z/VM
7.3

CMS Callable Services Reference

SC24-6259-73

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 771.

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2022-08-31

Contents

Figures

Tables

About This Document

This document describes the basic set of CMS callable services library (CSL) routines included with z/VM®. These routines are contained in the VMLIB callable services library. You should be able to use these CSL routines in programs.

The CSL routines provided in VMLIB contain:

- An overview of the VMLIB callable services and how to invoke them
- Detailed descriptions of the routines, arranged alphabetically
- Lists of the return codes and reason codes issued by VMLIB routines
- Information about binding files that must be used in programs that call certain routines
- Information about symbols defined in binding files for certain routines
- Descriptions of Information Names that can be used with the Extract/Replace routine to obtain or change specific subsets of system information

Intended Audience

You should be knowledgeable about programming and familiar with the CMS programming interfaces and the material covered in the *z/VM: CMS Application Development Guide*.

This information is for application programmers who work with high-level languages or assembler language and who want to use CMS file pools, the Extract/Replace facility, data integrity capabilities, VM data spaces, and other functions that these CSL routines provide. System programmers and IBM® system support personnel may also find this book useful.

Where to Find More Information

The following documents describe other VM callable services:

- *z/VM: CMS File Pool Planning, Administration, and Operation*
- *z/VM: CMS Application Multitasking*
- *z/VM: OpenExtensions Callable Services Reference*
- *Common Programming Interface Communications Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf)*
- *z/VM: DFSMS/VM Customization*
- *z/VM: DFSMS/VM Removable Media Services*

Other documents you may need to develop application programs are listed in the "Bibliography" on page 775.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to Send Your Comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

To send us your comments, go to z/VM Reader's Comment Form (https://www.ibm.com/systems/campaignmail/z/zvm/zvm-comments) and complete the form.

If You Have a Technical Problem

Do not use the feedback method. Instead, do one of the following:

- Contact your IBM service representative.
- Contact IBM technical support.
- See IBM: z/VM Support Resources (https://www.ibm.com/vm/service).
- Go to IBM Support Portal (https://www.ibm.com/support/entry/portal/Overview).

Summary of Changes for z/VM: CMS Callable Services Reference

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (**|**) to the left of the change.

SC24-6259-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Removal of the CMSDESK function, external GUI functions, and the GUICSLIB DCSS

The CMSDESK function, external GUI functions, and the GUICSLIB DCSS are removed. The CMS CMSDESK command, the CMS SET WORKSTATION command, and the CMS QUERY WORKSTATION command are no longer valid commands. References to CMS GUI are removed from the publications. References to the DMSWKST and WorkstationGetAddress command are removed.

SC24-6259-02, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

SC24-6259-00, z/VM 7.1 (September 2018)

This edition supports the general availability of z/VM 7.1.

Chapter 1. Invoking Callable Services

Most z/VM routines are stored in a callable services library (CSL). When a program contains a call to a CSL routine, the call is not resolved until the program is run and the call is made (as opposed to when the module is built).

Note: For general information about CMS programming interfaces, see the *z/VM: CMS Application Development Guide*.

z/VM comes with two callable services libraries:

- **VMLIB** contains routines that:
  - Call CMS file system management functions (CMS file pool and minidisk I/O)
  - Call CMS file pool administration functions
  - Access the current generation of REXX variables
  - Issue VM commands through a REXX exec
  - Call the CMS Extract/Replace facility, which enables application programs to obtain or modify selected system information without release or VM system dependencies
  - Manipulate the CMS program stack
  - Use the CMS Coordinated Resource Recovery (CRR) facility to maintain data integrity
  - Use VM data spaces
  - Call program-to-program communications functions using Common Programming Interface (CPI) Communications
  - Call CPI resource recovery functions
  - Provide CMS file pool exits

  This book describes the VMLIB routines in detail, except those for file pool exits, CPI Communications, and CPI resource recovery. The file pool exit routines are described in *z/VM: CMS File Pool Planning, Administration, and Operation*.

CPI Communications and CPI Resource Recovery Routines
Application programs must call CPI Communications and CPI resource recovery routines in a way that is different from other VMLIB routines. Routines that perform CPI Communications functions are documented in the *Common Programming Interface Communications Reference*. This includes common routines that can be used across various IBM systems, as well as VM extension routines. Routines that perform CPI resource recovery functions are documented in the *Common Programming Interface Resource Recovery Reference*. In addition, the *z/VM: CMS Application Development Guide* gives an overview of writing communications programs in VM, while *z/VM: CP Programming Services* provides additional reference information.

- **VMMTLIB** contains routines that:
  - Call CMS application multitasking functions
  - Call OpenExtensions (POSIX) services
  - Get the value set for the workstation display address

The CMS application multitasking routines are described in *z/VM: CMS Application Multitasking*. The routines for calling OpenExtensions services are described in the *z/VM: OpenExtensions Callable Services Reference*. The WorkstationGetAddress routine is described in this book.

DFSMS/VM provides two additional callable services libraries:

- **FSMPPSI** contains Removable Media Services (RMS) Tape Library Dataserver interface routines. These routines allow applications running under CMS OS simulation to issue requests for Tape Library Dataserver functions such as mounting or demounting a tape, querying library information, setting or resetting the device category, and setting the volume category. The routines are described in *z/VM: DFSMS/VM Removable Media Services*.
- **FSMPSI** contains DFSMS/VM installation-wide exit routines. These exits allow you to customize DFSMS/VM functions such as accounting records, authorization checking, migration and erasure verification, selecting the real device, authorized request pre-verification and post-verification, verifying shared resources, and issuing commands for device attachment and detachment. The routines are described in *z/VM: DFSMS/VM Customization*.

Loading VMLIB

All of the CSL routines described in this book (except WorkstationGetAddress) reside in VMLIB. Failure to load VMLIB can cause unpredictable results. The sample system profile (SYSPROF EXEC) loads VMLIB. If the call to RTNLOAD has been removed from SYSPROF EXEC, you can still have VMLIB loaded automatically by adding this line to your PROFILE EXEC:

```
RTNLOAD * (FROM VMLIB SYSTEM GROUP VMLIB)
```

VMLIB can take up to 1 MB of storage. For a small virtual machine, you may need to put VMLIB in a DCSS, increase the virtual machine size, or change SYSPROF EXEC to load only CSL routines critical to your applications.

Note: The WorkstationGetAddress routine resides in VMMTLIB. VMMTLIB is contained within the CMS nucleus and is automatically loaded during CMS initialization before the system profile is run.

Calling VMLIB CSL Routines

You can call VMLIB CSL routines from a program that is written in any of these programming languages:

- Assembler
- C
- COBOL (IBM COBOL II and OS/VS COBOL Program Products)
- VS FORTRAN
- VS Pascal
- PL/I
- REXX
- Ada

Calling Formats

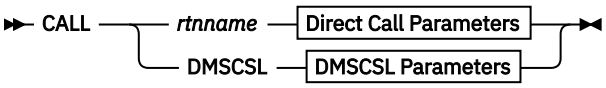
To call a CSL routine, whether it is a routine you created or one already in VMLIB, you must use one of these methods:

- Direct call
- Call to DMSCSL
- CSLFPI macro

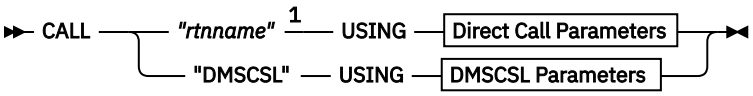
Any CSL routine is callable using CSLFPI, by a call to DMSCSL, or from REXX. See the individual command to determine whether it can be called directly. See the *z/VM: CMS Application Development Guide* for more information.

Here are the calling formats for all the supported languages:

General Format



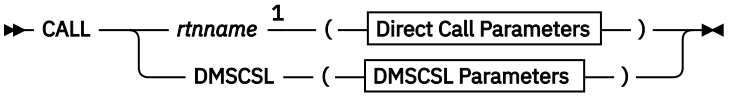
VS COBOL II



Notes:

[1] Do not use quotation marks on calls that use the longer names defined in binding files: `CALL StackQuery USING…`

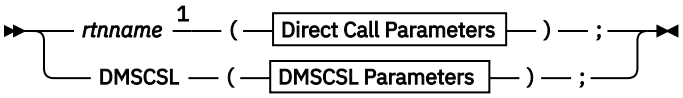
VS FORTRAN



Notes:

[1] VS FORTRAN cannot use the longer-style routine names, such as StackBufferCreate, to make direct calls. It must use the shorter name, such as DMSSTKC.

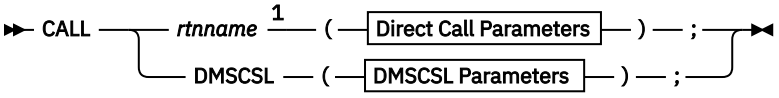
VS Pascal



Notes:

[1] VS Pascal cannot use the longer-style routine names, such as StackBufferCreate, to make direct calls. It must use the shorter name, such as DMSSTKC.

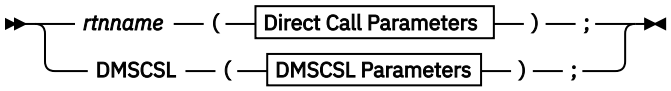
PL/I



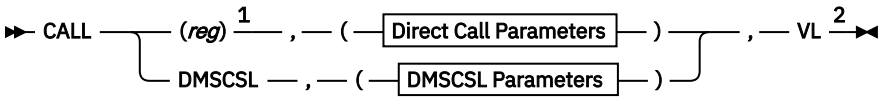
Notes:

[1] PL/I cannot use the longer-style routine names, such as StackBufferCreate, to make direct calls. It must use the shorter names, such as DMSSTKC.

C



Assembler



Notes:

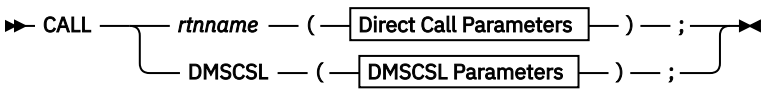
[1] When the long form of a routine's name is used in a direct call, the routine's address must be passed in a register. See example below.

[2] For the Assembler language, addresses used with CSL routines are 32-bit fields. The high-order bit is not used for addressing and must be zero, except that it must be set to one when it designates the end of a parameter list. Specifying VL on the routine call sets the high-order bit to 1. If you build the parameter list yourself and provide only the address of the list in the routine call, you must set the high-order bit of the last address in the list (see "Assembler Programs" on page 573).

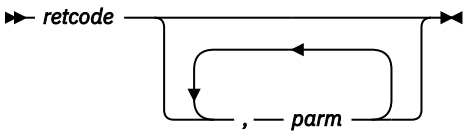
Example:

```
L     R15,=A(STACKQUERY)
CALL  (15),(RC,RE,BUFNUM,LINES,HIGH),VL
```

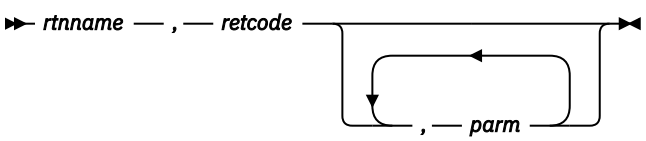
Ada



Direct Call Parameters

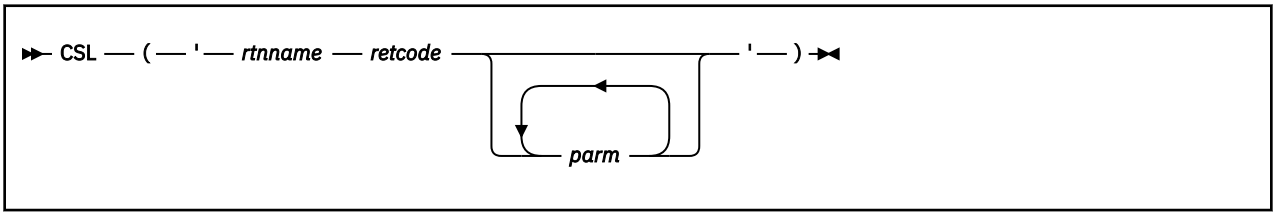


DMSCSL Parameters

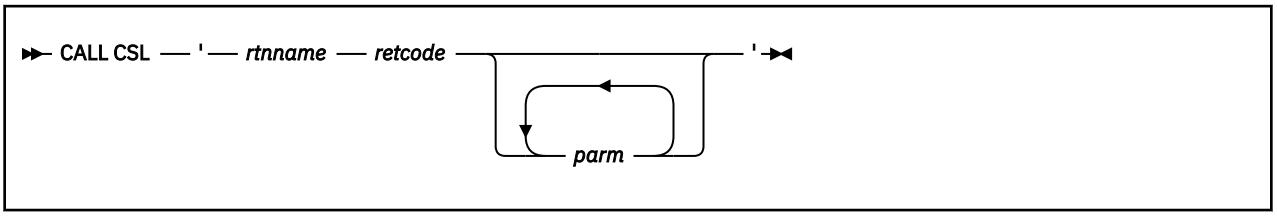


Calling CSL routines is different in REXX/VM. Routines can be called as functions or they can be called as routines with the CALL instruction or the ADDRESS OPENVM instruction:

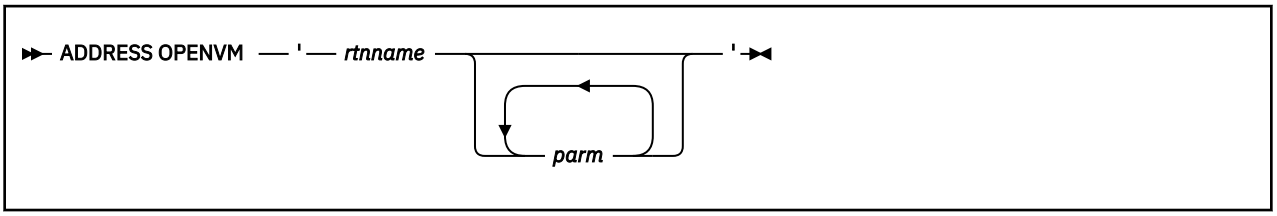
REXX function call



REXX CALL instruction



REXX ADDRESS OPENVM instruction



Note: OPENVM-type CSL routines can be called from REXX *only* by using the ADDRESS OPENVM interface. OPENVM routines may not follow the usual CSL conventions, such as providing return and reason codes as the first two parameters. For more information about the ADDRESS OPENVM interface, see the *z/VM: REXX/VM Reference*. To determine if a CSL routine is an OPENVM routine, you can use the CSLMAP or CSLLIST command. For information about these commands, see the *z/VM: CMS Commands and Utilities Reference*.

Call Parameters

retcode

passes back the return code from the CSL routine (it is also returned in register 15). This return code variable must be the second parameter in the DMSCSL parameter list, and the first parameter in the direct call parameter list. It must be a signed integer variable with a length of 4.

Return codes enable the calling program to maintain control in the event of a CSL error condition, such as a call to a routine that cannot be located at execution time, or a call with parameters of unacceptable data types.

Return codes are also generated when one of the interfaces encounters a problem, such as parameters not matching what is in the template file. Return codes from the interface are stored in the *retcode* parameter, but they are negative values. (Return codes from the ADDRESS OPENVM interface are returned in the REXX *RC* variable.)

For more information about return codes, see Appendix D, “Return Codes,” on page 597.

parm

specifies the name of a variable to be passed to the CSL routine. (Some programming languages let you pass literal values as parameters.) The parameters and their data types are described with the individual routines in this chapter.

Character strings can be specified in lowercase, mixed case, or uppercase. In most instances, the characters are translated into uppercase. For file identifiers, however, the file name and file type are **not** translated to uppercase; they remain in the case in which they were specified.

See the *z/VM: CMS Application Development Guide* for examples of calling CSL routines from programs written in high-level languages. For more information about using the REXX CSL interface, see the *z/VM: REXX/VM Reference*.

Programming Language Binding Files

Programs that use CSL routines with longer names like "StackBufferCreate", rather than short names like "DMSERP", need to include programming language binding files. Binding files declare external functions, constants, and return and reason codes. In the descriptions of these routines, the values that can be assigned to parameters and the values of return and reason codes are given by the symbols defined in the binding files. For more information, see Appendix A, "Using Programming Language Binding Files," on page 571 and Appendix B, "Symbols Defined in Binding Files," on page 575.

Overview of VMLIB CSL Routines by Task

The following sections list the VMLIB CSL routines divided according to task. The principal task categories are:

- Calling a REXX exec
- Accessing REXX variables from a program
- Using VM data spaces
- Using the CMS Extract/Replace function
- Manipulating the CMS program stack
- File system management (file pool and minidisk I/O)
- File pool administration
- Error checking and debugging
- Resource recovery
- Miscellaneous

For complete information about how to code your own callable services library routines and how to make your own libraries, see the *z/VM: CMS Application Development Guide for Assembler*.

Calling a REXX Exec

You can use CMS commands from a REXX exec by using the DMSCCE CSL routine.

Table 1. Routine for Calling a REXX Exec

Routine Name	Description
DMSCCE	Call a REXX exec from an application program. For more information, refer to "DMSCCE - Call a REXX Exec" on page 30.

Accessing REXX Variables from a Program Written in Another Language

A program called from a REXX exec can use CSL routines to access and manipulate variables of the calling exec. These CSL routines must be called from a program that is not written in REXX.

Table 2. Routines for Accessing REXX Variables

Routine Name	Description
DMSCDR	Drop a REXX variable. For more information, refer to "DMSCDR - Drop a REXX Variable" on page 33.
DMSCGR	Get the value of a REXX variable. For more information, refer to "DMSCGR - Get a REXX Variable" on page 35.
DMSCGS	Get the value of one of the special REXX variables. For more information, refer to "DMSCGS - Get Special REXX Values" on page 37.

Table 2. Routines for Accessing REXX Variables (continued)	
Routine Name	**Description**
DMSCGX	Get the value of the next stored REXX variable. For more information, refer to "DMSCGX - Get the Next REXX Variable" on page 39.
DMSCSR	Assign a value to a REXX variable. For more information, refer to "DMSCSR - Set a REXX Variable" on page 116.

Your program does not have to be concerned with how the variables are accessed, how storage is handled, or any other internals—REXX/VM takes care of it all.

(These CSL routines work using the EXECCOMM interface, which is described in detail in the *z/VM: REXX/VM Reference*.)

VM Data Space Routines

This section describes routines for creating and using VM data spaces in an XC virtual machine. A user with an XA virtual machine can use data space services to share primary address space or copy data from another user's address space into his own primary address space. The CSL routines for using data spaces have the same syntax conventions as the file system (I/O) routines.

Table 3. Routines for Using VM Data Spaces	
Routine Name	**Description**
DMSSPCC	Creates a data space for an XC virtual machine; when called from an XA virtual machine, the primary address space is made shareable. For more information, refer to "DMSSPCC - Create Data Space" on page 484.
DMSSPCD	Deletes a data space for an XC virtual machine; when called from an XA virtual machine, only the data space structures related to the primary address space are deleted. For more information, refer to "DMSSPCD - Delete Data Space" on page 487.
DMSSPCI	Isolates address space from other users and programs. For more information, refer to "DMSSPCI - Isolate Address Space" on page 489.
DMSSPCP	Permits another user or program to access an address space. For more information, refer to "DMSSPCP - Permit Address Space Access" on page 491.
DMSSPCPY	Allows an XA virtual machine to copy information from an address space. For more information, refer to "DMSSPCPY - Copy from Address Space" on page 495.
DMSSPCQ	Extracts information about an address space that a virtual machine owns or is authorized to access. For more information, refer to "DMSSPCQ - Query Address Space" on page 497.
DMSSPCR	Restores other users' access to a virtual machine's shared address space that had been isolated with a call to DMSSPCI. For more information, refer to "DMSSPCR - Restore Address Space Access" on page 499.
DMSSPCRP	Releases specified pages of an address space. For more information, refer to "DMSSPCRP - Release Address Space Pages" on page 501.
DMSSPLA	Establishes addressability to an address space. For more information, refer to "DMSSPLA - Establish Address Space Addressability" on page 503.
DMSSPLR	Removes addressability to an address space. For more information, refer to "DMSSPLR - Remove Address Space Addressability" on page 506.

CMS Extract/Replace Function

You may obtain or modify selected system information without release or VM system dependencies by using the following CSL routines. For examples of calling the Extract/Replace function from various languages, see the *z/VM: CMS Application Development Guide*.

Table 4. Extract/Replace Routine	
Routine Name	**Description**
DMSERP	EXTRACT - Obtains system information. For more information, refer to "DMSERP - Extract/Replace" on page 160.
DMSERP	REPLACE - Updates system information. For more information, refer to "DMSERP - Extract/Replace" on page 160.
DMSERP	RESET - Initializes Extract/Replace. For more information, refer to "DMSERP - Extract/Replace" on page 160.

Program Stack Routines

CSL routines can read and write to the program stack, add and drop CMS program stack buffers, and query the number of lines and number of buffers on the program stack. The routines can be called directly or through the DMSCSL interface. They require language binding files and use symbols for return and reason codes (see Appendix A, "Using Programming Language Binding Files," on page 571).

Table 5. Program Stack Routines	
Routine Name	**Description**
StackBufferCreate	Adds a buffer to the program stack. For more information, refer to "StackBufferCreate / DMSSTKC - Add a Buffer to the Program Stack" on page 558.
StackBufferDelete	Drops buffers from the program stack. For more information, refer to "StackBufferDelete / DMSSTKD - Drop Buffers from the Program Stack" on page 560.
StackQuery	Queries the number of lines and the number of buffers in the program stack. For more information, refer to "StackQuery / DMSSTKQ - Query the Program Stack" on page 562.
StackRead	Pops a line from the program stack. For more information, refer to "StackRead / DMSSTKR - Read from the Program Stack" on page 564.
StackWrite	Pushes or queues a line to the program stack. For more information, refer to "StackWrite / DMSSTKW - Write to the Program Stack" on page 567.

File System Management (File Pool and Minidisk I/O) Routines

This section describes general routines for manipulating data stored in CMS file pools or on minidisks. These file system management routines are available to programmers writing applications for the CMS environment. The routines listed in this section adhere to the syntax conventions described later in this chapter. Many of these routines operate on both file pool and minidisk files.

The routines that operate on file pool data are primarily intended for manipulating CMS Shared File System (SFS) files and directories. However, many of these routines also provide limited support for manipulating OpenExtensions Byte File System (BFS) files and directories. This support is primarily intended for file pool administration tasks and system-managed storage. The primary CSL interface for manipulating BFS files and directories is the set of routines described in the *z/VM: OpenExtensions Callable Services Reference*.

For more information about file system management, see the *z/VM: CMS Application Development Guide*. For information about SFS and related topics, see *z/VM: CMS File Pool Planning, Administration, and Operation*. For information about BFS, see the *z/VM: OpenExtensions User's Guide*.

Basic File Management Routines

The basic file management routines are listed and defined in the following table.

Table 6. Basic File Management Routines	
Routine Name	**Description**
DMSCLDBK	Closes files previously opened by DMSOPDBK. For more information, refer to "DMSCLDBK - Close Data Block" on page 57.

Table 6. Basic File Management Routines (continued)

Routine Name	Description
DMSCLOSE	Closes files previously opened by DMSOPEN. For more information, refer to "DMSCLOSE - Close" on page 66.
DMSCRALI	Creates an alias of a file in an SFS directory. For more information, refer to "DMSCRALI - Create Alias" on page 81.
DMSCRFIL	Creates a new file in an SFS directory without providing any data for the file. For more information, refer to "DMSCRFIL - Create File" on page 95.
DMSCROB	Creates an external object entry in an SFS directory. For more information, refer to "DMSCROB - Create External Object" on page 109.
DMSEXIFI	Checks for the existence of a file and returns information in parameters. For more information, refer to "DMSEXIFI - Exist - File" on page 179.
DMSFILEC	Copies one SFS or minidisk file to another SFS or minidisk file. For more information, refer to "DMSFILEC - Filecopy" on page 206.
DMSOPDBK	Makes a file usable for subsequent reading or writing data blocks. For more information, refer to "DMSOPDBK - Open Data Block" on page 319.
DMSOPEN	Makes a file usable for subsequent reading or writing records. For more information, refer to "DMSOPEN - Open" on page 340.
DMSPOINT	Adjusts the read and write pointers in a file opened by DMSOPEN. For more information, refer to "DMSPOINT - Point" on page 361.
DMSQOBJ	Returns the Remote Name of an external object. For more information, refer to "DMSQOBJ - Query External Object" on page 402.
DMSRDDBK	Gets one or more blocks from a file opened by DMSOPDBK. For more information, refer to "DMSRDDBK - Read Data Block" on page 436.
DMSREAD	Gets one or more records from a file opened by DMSOPEN. For more information, refer to "DMSREAD - Read" on page 439.
DMSTRUNC	Truncates an SFS or minidisk file to a specified record. For more information, refer to "DMSTRUNC - Truncate" on page 521.
DMSVALDT	Validates an input file ID. For more information, refer to "DMSVALDT - Validate" on page 533.
DMSWRDBK	Writes one or more blocks to a file opened by DMSOPDBK. For more information, refer to "DMSWRDBK - Write Data Block" on page 544.
DMSWRITE	Writes one or more records to a file opened by DMSOPEN. For more information, refer to "DMSWRITE - Write" on page 547.

SFS File Attribute Control Routines

The routines for controlling SFS file attributes are listed and defined in the following table.

Table 7. SFS File Attribute Control Routines

Routine Name	Description
DMSCATTR	Modifies the recoverability and overwrite attributes of the specified SFS file. For more information, refer to "DMSCATTR - Change Attributes" on page 25.
DMSPOPA	Removes any recoverability and overwrite attributes specified previously with the DMSPUSHA routine. For more information, refer to "DMSPOPA - Pop Attribute" on page 364.
DMSPUSHA	Defines a set of recoverability and overwrite attributes for each file mode number. For more information, refer to "DMSPUSHA - Push Attributes" on page 370.

General Routines for Managing Files and Directories

The CSL routines that can be used for either files or directories are listed and defined in the following table.

Table 8. General CSL Routines for Managing Files and Directories	
Routine Name	**Description**
DMSCRLOC	Creates an explicit lock on an SFS file or directory or a BFS regular file. For more information, refer to "DMSCRLOC - Create Lock" on page 102.
DMSDELOC	Releases an explicit lock created by DMSCRLOC. For more information, refer to "DMSDELOC - Delete Lock" on page 118.
DMSERASE	Erases files, aliases, external objects, and SFS directories. For more information, refer to "DMSERASE - Erase" on page 152.
DMSEXIST	Checks for the existence of a file, directory, or external object. For more information, refer to "DMSEXIST - Exist" on page 191.
DMSGRANT	Grants authority to another user to use an SFS file or directory. For more information, refer to "DMSGRANT - Grant Authority" on page 283.
DMSRELOC	Moves an SFS file, external object, or a directory subtree from one directory to another. For more information, refer to "DMSRELOC - Relocate" on page 455.
DMSRENAM	Renames a mindisk file, SFS file, alias, external object, or SFS subdirectory. For more information, refer to "DMSRENAM - Rename" on page 460.
DMSREVOK	Removes user authority from an SFS file or directory. For more information, refer to "DMSREVOK - Revoke Authority" on page 469.

Basic Directory Management Routines

The basic directory management routines are listed and defined in the following table.

Table 9. Basic Directory Management Routines	
Routine Name	**Description**
DMSCLDIR	Closes a directory opened by DMSOPDIR. For more information, refer to "DMSCLDIR - Close Directory" on page 63.
DMSCRDIR	Creates a new SFS directory. For more information, refer to "DMSCRDIR - Create Directory" on page 88.
DMSDIRAT	Sets or removes the directory attribute for an SFS directory. For more information, refer to "DMSDIRAT - Set Directory Attribute" on page 126.
DMSEXIDI	Checks for the existence of an SFS or BFS directory and returns information in parameters. For more information, refer to "DMSEXIDI - Exist - Directory" on page 171.
DMSGETDA	Reads one SFS directory record after the directory has been opened with an intent of SEARCHALL and returns information in parameters. For more information, refer to "DMSGETDA - Get Directory - Searchall" on page 214.
DMSGETDD	Reads one directory record after the directory has been opened with an intent of DIR and returns information in parameters. For more information, refer to "DMSGETDD - Get Directory - Dir" on page 219.
DMSGETDF	Reads one SFS or minidisk directory record after the directory has been opened with an intent of FILE and returns information in parameters. For more information, refer to "DMSGETDF - Get Directory - File" on page 222.
DMSGETDI	Reads directory records and returns information in a buffer. For more information, refer to "DMSGETDI - Get Directory" on page 228.
DMSGETDK	Reads one SFS or BFS directory record after the directory has been opened with an intent of LOCK and returns information in parameters. For more information, refer to "DMSGETDK - Get Directory - Lock" on page 243.
DMSGETDL	Reads one SFS directory record after the directory has been opened with an intent of ALIAS and returns information in parameters. For more information, refer to "DMSGETDL - Get Directory - Alias" on page 247.
DMSGETDS	Reads one SFS directory record after the directory has been opened with an intent of SEARCHAUTH and returns information in parameters. For more information, refer to "DMSGETDS - Get Directory - Searchauth" on page 251.

Table 9. Basic Directory Management Routines (continued)	
Routine Name	**Description**
DMSGETDT	Reads one SFS directory record after the directory has been opened with an intent of AUTH and returns information in parameters. For more information, refer to "DMSGETDT - Get Directory - Auth" on page 257.
DMSGETDX	Reads one SFS or BFS directory record after the directory has been opened with an intent of FILEEXT and returns information in parameters. For more information, refer to "DMSGETDX - Get Directory - File Extended" on page 262.
DMSOPDIR	Allows a directory to be used by any of the Get Directory routines. For more information, refer to "DMSOPDIR - Open Directory" on page 332.

File Pool Administration Routines

Table 10 on page 11 lists routines available for performing programming tasks associated with file pool administration. For more information about file pool administration, see *z/VM: CMS File Pool Planning, Administration, and Operation*.

Table 10. File Pool Administration Routines	
Routine Name	**Description**
DMSCLBLK	Closes files previously opened by DMSOPBLK. For more information, refer to "DMSCLBLK - Close Blocks" on page 47.
DMSCLCAT	Closes the user storage group, file space, or directory opened by DMSOPCAT. For more information, refer to "DMSCLCAT - Close Catalog" on page 54.
DMSCPYBF	Obtains a buffer large enough to hold all the information previously returned by DMSOPCAT or DMSCLCAT. For more information, refer to "DMSCPYBF - Copy Buffer" on page 79.
DMSDEUSR	Removes one or more file spaces from a file pool. For more information, refer to "DMSDEUSR - Delete File Space" on page 122.
DMSDISFS	Places a disable lock on a file space. For more information, refer to "DMSDISFS - Disable File Space" on page 130.
DMSDISSG	Places a disable lock on a storage group. For more information, refer to "DMSDISSG - Disable Storage Group" on page 134.
DMSENAFS	Releases a disable lock on a file space. For more information, refer to "DMSENAFS - Enable File Space" on page 138.
DMSENASG	Releases a disable lock on a storage group. For more information, refer to "DMSENASG - Enable Storage Group" on page 142.
DMSENUSR	Enrolls a file space in a file pool. For more information, refer to "DMSENUSR - Enroll File Space" on page 145.
DMSOPBLK	Makes a file usable for reading blocks or writing blocks. For more information, refer to "DMSOPBLK - Open Blocks" on page 298.
DMSOPCAT	Opens a user storage group or file space to read or write catalog information, or opens a directory to read catalog information. For more information, refer to "DMSOPCAT - Open Catalog" on page 313.
DMSQFPDD	Returns one record of information from the buffer created by DMSQFPDS. For more information, refer to "DMSQFPDD - Query File Pool Disable - Deblocker" on page 385.
DMSQFPDS	Returns information about locked storage groups and file spaces. For more information, refer to "DMSQFPDS - Query File Pool Disable" on page 388.
DMSQLIMA	Returns limits information (in a buffer) about all file spaces enrolled in a file pool. For more information, refer to "DMSQLIMA - Query Limits" on page 394.
DMSQLIMD	Returns limits information for one file space from the buffer returned by DMSQLIMA. For more information, refer to "DMSQLIMD - Query Limits - Deblocker" on page 397.
DMSQLIMU	Returns limits information for one file space. For more information, refer to "DMSQLIMU - Query Limits - Single File Space" on page 399.
DMSQUSG	Returns information about user storage groups in a file pool. For more information, refer to "DMSQUSG - Query User Storage Group" on page 409.

Table 10. File Pool Administration Routines (continued)	
Routine Name	**Description**
DMSQUSGD	Returns the information from DMSQUSG in deblocked form. For more information, refer to "DMSQUSGD - Query User Storage Group - Deblocker" on page 412.
DMSRDBLK	Obtains one or more blocks from a file. For more information, refer to "DMSRDBLK - Read Blocks" on page 415.
DMSRDCAT	Returns catalog information. For more information, refer to "DMSRDCAT - Read Catalog" on page 418.
DMSRELBK	Releases alternate blocks in a storage group. For more information, refer to "DMSRELBK - Release Blocks" on page 453.
DMSWRACC	Writes file pool server accounting records. For more information, refer to "DMSWRACC - Write File Pool Server Accounting Records" on page 536.
DMSWRBLK	Writes one or more blocks to a file opened by DMSOPBLK. For more information, refer to "DMSWRBLK - Write Blocks" on page 538.
DMSWRCAT	Writes catalog information to storage group catalogs opened by DMSOPCAT. For more information, refer to "DMSWRCAT - Write Catalog" on page 541.

Error Checking and Debugging Routines

This section describes routines available to the CMS user for getting error information and to aid in debugging programs. The routines listed and described in the following table have the same syntax conventions as the file system management routines, although they are of more general use.

Table 11. Error Checking and Debugging Routines	
Routine Name	**Description**
DMSGETSP	Iteratively retrieves all error blocks for all warnings and errors detected since the last time the work unit was committed or rolled back. For more information, refer to "DMSGETSP - Get Synchronization Point Errors" on page 277.
DMSPCAER	Parses error information returned by the CMS Protected Conversation Adapter (PCA) during synchronization point processing. For more information, refer to "DMSPCAER - Protected Conversation Adapter Errors" on page 355.
DMSTRACE	Allows applications to record trace data. For more information, refer to "DMSTRACE - Trace" on page 517.
DMSWUERR	Converts SFS *wuerror* output into individual variables. For more information, refer to "DMSWUERR - Work Unit Error Data Deblocker" on page 553.

Resource Recovery-Related Routines

This section describes CSL routines available to the CMS user for protecting data integrity through the use of Coordinated Resource Recovery (CRR). This facility ensures that an application program can update several resources with integrity. Also included in this section are routines for managing CMS work units.

The routines listed in tables Table 12 on page 12 and Table 13 on page 13 have the same syntax conventions as the file system management routines, although they are of more general use. You can find a sample program that uses some of these routines in the *z/VM: CMS Application Development Guide*.

Coordinated Resource Recovery Routines

Table 12. Coordinated Resource Recovery Related Routines	
Routine Name	**Description**
DMSCOMM	Commits all changes to protected resources (such as SFS files and directories) on the work unit. For more information, refer to "DMSCOMM - Commit" on page 71.
DMSLUWID	Returns the logical unit of work ID associated with a work unit. For more information, refer to "DMSLUWID - Get a Logical Unit of Work ID" on page 294.

Table 12. Coordinated Resource Recovery Related Routines (continued)	
Routine Name	**Description**
DMSROLLB	Removes all uncommitted changes for a work unit since the last commit. For more information, refer to "DMSROLLB - Rollback" on page 477.
DMSSETAG	Lets an application provide a transaction tag to aid recovery from a failure. For more information, refer to "DMSSETAG - Set Transaction Tag" on page 480.
DMSSSPTO	Lets an application set synchronization point options for CRR. For more information, refer to "DMSSSPTO - Set Synchronization Point Options" on page 508.

Work Unit Management Routines

Table 13. Work Unit Management Routines	
Routine Name	**Description**
DMSGETWU	Obtains a unique work unit ID. For more information, refer to "DMSGETWU - Get Work Unit ID" on page 280.
DMSPOPWU	Removes a work unit ID from the work unit ID stack. For more information, refer to "DMSPOPWU - Pop Default Work Unit ID" on page 366.
DMSPURWU	Gives back all work units. For more information, refer to "DMSPURWU - Purge Work Unit IDs" on page 368.
DMSPUSWU	Makes a work unit ID the default work unit ID. For more information, refer to "DMSPUSWU - Push Default Work Unit ID" on page 373.
DMSQWUID	Returns the current default work unit identifier. For more information, refer to "DMSQWUID - Query Work Unit ID" on page 414.
DMSRETWU	Indicates to CMS that your application has completed all work on the specified work unit. For more information, refer to "DMSRETWU - Return Work Unit ID" on page 466.

CRR Participation Routines

This section lists routines for enabling a resource manager to participate in CRR.

Table 14. CRR Participation Routines	
Routine Name	**Description**
DMSCHREG	Changes certain registry values for a resource registered with the synchronization point manager. For more information, refer to "DMSCHREG - CRR Change Registration" on page 43.
DMSGETER	Retrieves error blocks for warnings and error that the resource adapter has detected. For more information, refer to "DMSGETER - CRR Get My Errors" on page 271.
DMSGETRS	Obtains the CRR recovery server's current log name and transaction program name. For more information, refer to "DMSGETRS - CRR Get Recovery Server Information" on page 275.
DMSMARK	Marks the completion of an asynchronous event. For more information, refer to "DMSMARK - CRR Mark Request ID" on page 296.
DMSREG	Registers a protected resource and its resource adapter with the CRR sync point manager. For more information, refer to "DMSREG - CRR Resource Adapter Registration" on page 444.
DMSSETR	Tells the sync point manager that a backout is required for the work unit. For more information, refer to "DMSSETR - CRR Set Received" on page 482.
DMSUNREG	Removes an instance of registration from the sync point manager coordination list. For more information, refer to "DMSUNREG - CRR Resource Adapter Unregistration" on page 531.

Miscellaneous CSL Routines

This section describes routines available to the CMS user for miscellaneous programming tasks. The routines listed and described in the following table have the same syntax conventions as the file system management routines, although they are of more general use.

Table 15. Other CSL Routines

Routine Name	Description
DMSCALLR	Examines the command call chain. For more information, refer to “DMSCALLR - Get Caller Identification” on page 24.
DMSCHECK	Determines if an SFS or BFS asynchronous request has been completed. For more information, refer to “DMSCHECK - Check” on page 41.
DMSCPR	Interfaces with system Data Compression Services. For more information, refer to “DMSCPR - Data Compression Services” on page 76.
DMSESM	Identifies the application to an External Security Manager (ESM). For more information, refer to “DMSESM - Identify Program to External Security Manager” on page 169.
DMSGETFM	Returns the first free file mode letter. For more information, refer to “DMSGETFM - Get File Mode” on page 274.
DMSLINK	Obtains a link to a user's minidisk or virtual reader queue. For more information, refer to “DMSLINK - Link to User Minidisk or Virtual Reader Queue” on page 291.
DMSPWCHK	Verifies that a user ID and password are valid; optionally verifies a user's LOGON BY privileges to another user. For more information, refer to “DMSPWCHK - Verify Logon Password” on page 374.
DMSQCONN	Determines if a file pool server is set at a local or remote location. For more information, refer to “DMSQCONN - Query Connect” on page 376.
DMSQEFL	Returns codes indicating the CP and CMS functional levels. For more information, refer to “DMSQEFL - Query Functional Level of CP and CMS” on page 379.
DMSQFMOD	Determines whether an SFS directory or minidisk is accessed with a specified file mode. For more information, refer to “DMSQFMOD - Query File Mode” on page 383.
DMSQSFSL	Returns information about the functional level of a file pool server. For more information, refer to “DMSQSFSL - Query File Pool Server Level” on page 406.
DMSUDATA	Sends information to an external security manager. For more information, refer to “DMSUDATA - Send User Data” on page 528.
DTCXLATE	Obtains information from a TCP/IP translation table file. For more information, refer to “DTCXLATE - Read TCP/IP Translation Table” on page 556.
WorkstationGetAddress	Gets the value set for the workstation display address.

Syntax Conventions for CSL Routines

This section describes syntax conventions that apply to most CSL routines.

For each routine, the first parameter following the routine name is always a signed integer return code (*retcode*). The second parameter following the routine name is often a signed integer reason code (*reascode*). Signed integer parameters always have a length of 4 bytes.

Compound Variables

Some character parameters do not have a fixed length or they are compound and have blanks before and after the constituent strings. For example, one of DMSCRLOC's parameters consists of a keyword from each of two groups: 1) SHARE, UPDATE or EXCLUSIVE and 2) SESSION or LASTING. Each field of a compound input parameter is discussed in the *Parameters* section of the routine description. These separate fields must be combined into a compound character parameter for inclusion in the parameter list of the CSL routine. The combined length of these fields, including any leading, trailing, or intervening blanks, must then be specified in the length parameter that follows.

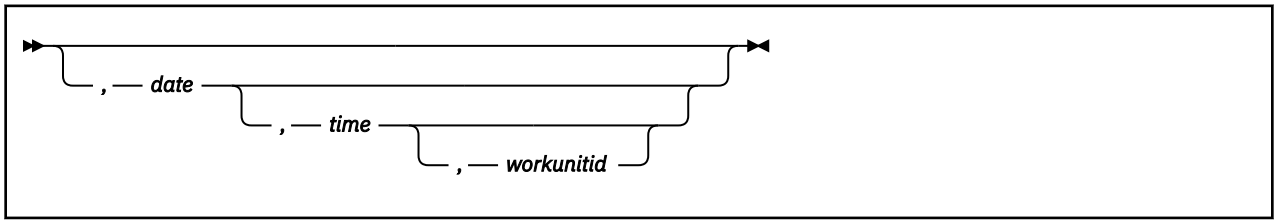
The following example (using REXX syntax) illustrates how such a compound parameter might be coded, assuming that *datafile* and *datadir* have been defined as namedefs:

```
retcode=0
reascode=0
fileid = datafile datadir
length1 = length(fileid)
opts.1 = 'UPDATE'
opts.2 = 'SESSION'
opts = opts.1 opts.2
length2 = length(opts)

call csl 'DMSCRLOC retcode reascode fileid length1 opts length2'
```

Optional Parameters

When the last list of parameters is optional and you do not want to specify it as an option, you may stop the list after the last required parameter or the last optional parameter that is specified. Many of these routines include several optional parameters at the end of their parameter lists. If you want to use the last optional parameter listed, however, you cannot simply skip the optional parameters preceding it. You must specify all preceding optional parameters. These parameters typically appear in the syntax diagram of a routine like this:



In this example, if you want to use the *workunitid* parameter, you must also specify the *date* and *time* parameters. If you want to use only the *time* parameter, you can omit *workunitid* but you must still specify *date*.

Common Parameters

In addition to the *retcode* parameter (described in page "Call Parameters" on page 5), the following parameters are used by many of the file system management and related routines:

reascode
is a 5-digit reason code associated with the return code placed in the *retcode* parameter. (The reason code is also returned in general register 0.) This reason code provides further information about the error or warning. See "Common Reason Codes" on page 601 for a table of reason codes that can occur with many file system management (I/O) routines and other related routines. Reason codes specific to a routine are documented at the end of the routine description.

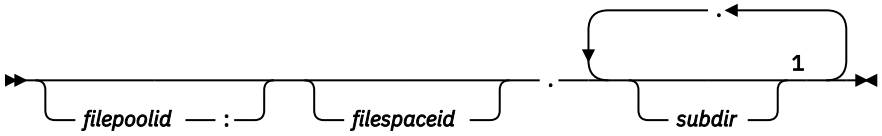
fn_ft
is a file name and file type. For some routines it is required and for some it is optional. This is a character string that contains up to 8 characters for the file name followed by up to 8 characters for

the file type. There must be at least one blank between the file name and the file type. This character string can be in mixed case. If it is, CMS does not convert it to uppercase as it does with other character strings. The mixed case differences are retained as a distinguishing feature of the strings.

For a BFS file, *fn* and *ft* are system-generated values that together uniquely identify the file within a byte file system.

When *fn* and *ft* are specified in any routine without further explanation, they are the usual CMS file name and file type with the standard syntax.

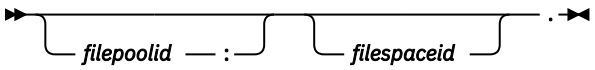
dirname
is the name of an SFS directory. This is a character string in the form:



Notes:

[1] Up to eight subdirectories can be specified.

bfsid
is the name of a byte file system. This is a character string in the form:



filepoolid
is the name of a file pool that the request is destined for. If not specified, the default file pool that you set with the SET FILEPOOL command is used (the system does not supply a default). You (or the system administrator) can also set a default file pool in your user CP directory so you do not have to enter the SET FILEPOOL command each time you log on. The file pool name can be up to 8 characters long. The first character must be alphabetic, but the remaining characters can be alphabetic or numeric. Lowercase is converted to uppercase.

: (colon)
is a separator that must be specified following the *filepoolid* when it is part of a directory name or byte file system name.

filespaceid
is the name of a file space (top directory). In an SFS directory name, this is the user ID of the owner of the directory. The file space ID can be up to 8 characters long. Lowercase is converted to uppercase. It defaults first to the file space ID set with the SET FILESPACE command, and then to the user ID calling the routine.

. (period)
is a separator that must be specified following the *filespaceid* and between *subdir* names. If it is specified without the *filepoolid* and *filespaceid* parameters, it means the top directory in the default file space in the default file pool.

subdir
is the name of a subdirectory. A subdirectory name can be up to 16 characters long. It can contain alphabetic or numeric characters, and the following special characters: @, #, _, $. Lower case is converted to uppercase.

filemode
is the file mode letter of an accessed minidisk or SFS directory. It is a character variable with a length of 1 and contains a value from A-Z or asterisk. An asterisk is used by some routines to search for the first occurrence of a file in the CMS search order.

fmnumber
is the numeric part of the file mode. It is a character variable with a length of 1 and with a value between 0 and 6, inclusive. The file mode number is only meaningful for files. Do not use it when referring to a directory. If not specified and no additional information is included with the routine, the default is 1.

namedef
is a temporary name that represents a *fn_ft*, *dirname*, *bfsid*, or *filemode*. This is a character variable with a length between 1 and 16. It is created before a program starts by entering the CREATE NAMEDEF command. The namedef can be used by a program instead of coding the specific file name and file type, SFS directory name, byte file system name, or file mode letter.

token
is an 8-byte character variable used to identify an instance of an open file or directory. It is passed from the routine to your program on DMSOPEN, DMSOPBLK, DMSOPCAT, DMSOPDBK, and DMSOPDIR. Specify the token on subsequent Read, Write, Get, and Close routines for the same file or directory.

Note: The use of some of these routines requires file pool administration or other authority.

workunitid
identifies a work unit associated with a group of selected operations. This is a signed integer variable with a length of 4. You can specify the same work unit ID on routines that operate on different file pools or other resources.

When a function is operating on a minidisk file, no work is associated with a work unit. The work unit in this case refers solely to previous work done using this work unit. The work unit groups operations to SFS file pools and operations using APPC/VM protected conversations. If, for example, you erase a minidisk file using DMSERASE and issue the COMMIT parameter, any work that is still uncommitted on the work unit will then be committed. Specifying COMMIT or NOCOMMIT in this case had no effect on when the minidisk file was actually erased.

Divide your work into logical units of work, that is, sets of changes that need to be committed as a unit. All changes associated with a work unit since either the beginning of the work unit or since the last commit (or rollback) get committed (or rolled back) at the same time. After committing a work unit, you can use the same work unit ID to identify the next logical unit of work. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

requestid
identifies a specific asynchronous request. It is a 4-byte integer field. If it is omitted or contains a binary 0 on input, the request is handled synchronously. If it contains a binary 1 on input, the request is handled asynchronously, and CMS returns an integer in *requestid* to identify the asynchronous request on a later Check (DMSCHECK) request. CMS returns the input value of 1 if the request is completed synchronously for any reason. For instance, all requests for minidisk files are processed synchronously, even if the input in *requestid* asks for an asynchronous request.

You cannot call CSL routines asynchronously from a REXX program.

`COMMIT`
means to keep changes. This is a character variable with a length of 6. Specifying the COMMIT option results in the coordinated commit of all protected resources updated on the work unit. If your application accesses other resources that do not participate in CRR, it must commit those resources using the commit interface of those resources. Once committed, these changes are permanent. They cannot be rolled back.

All changes associated with the work unit are committed when the routine finishes. That is, all changes made during a work unit, from either the start of the work unit or from the last commit or rollback, are kept. This includes changes to more than one file pool and other protected resources on the same work unit. If NOCOMMIT is specified, changes are not committed until DMSCOMM is issued, or the COMMIT parameter for the same work unit on another routine is specified, or until successful end-of-command, when CMS implicitly commits the changes.

The COMMIT and NOCOMMIT parameters do not affect minidisks. Many CSL routines that operate on minidisk and SFS files (such as DMSCLOSE) require you to specify whether you want to commit the changes to the work unit or leave them in an uncommitted state (NOCOMMIT). For operations on minidisk files, the COMMIT and NOCOMMIT parameters refer solely to the work unit and have no effect on when the changes to a minidisk are committed. Work done to a minidisk is not associated with a work unit. For example, if you create a minidisk file using DMSOPEN and DMSWRITE, you must close the file using DMSCLOSE. The file will be committed after it is closed (unless there are other files on that minidisk that remain open for output). If COMMIT is specified on DMSCLOSE, all uncommitted work associated with the work unit that was specified when the minidisk file was opened (DMSOPEN), is committed. If NOCOMMIT is specified, the work on the work unit remains unchanged.

When multiple protected resources are involved in a commit and a failure occurs, CMS either waits for CRR resynchronization to finish before returning to the application, or does not wait, depending upon the setting of the synchronization point options. These options may be set using "DMSSSPTO - Set Synchronization Point Options" on page 508.

A return code of 8 from a routine with the COMMIT parameter specified means that the changes made on the work unit were neither committed nor rolled back: the work remains uncommitted. With one exception, a return code of 8 means the operation could not be performed. However, if reason code 50500 is also returned, it means the operation was completed but the work unit could not be committed, usually because of an attempt to exceed the user's allotted file space. If an operation finishes successfully for a minidisk file, the COMMIT operation may still fail for the work unit. In this case, a warning return code is returned with the reason code explaining the exact cause of the error. If *wuerror* is specified, when reason code 50500 is returned the *error_reascode_info* field on one of the FPERROR entries contains a code for the specific reason that the work unit was not committed.

If a COMMIT request is issued and the file or directory is open, uncommitted updates are committed and the file or directory remains open. However, the following restrictions apply:

- Catalogs cannot be open on the work unit at commit time as the result of an Open Catalog request.
- Open files that have been modified through the block level interfaces (DMSWRBLK, DMSWRDBK) prevent a commit. Note that specifying the COMMIT option on when you close such a file is acceptable when it is the only file that is currently open using these interfaces.
- The commit request will fail if either the CMS user machine or any file pool server machine with open files on the work unit is not able to commit files while they are open.
- Atomic requests cannot be issued if there is outstanding work in the affected file pool for the work unit on which they are issued. Therefore, you cannot issue an atomic request even after a COMMIT when files are still open in the file pool for the work unit.

Certain errors can cause the work unit to be rolled back. A rollback causes all uncommitted work on the work unit to be discarded. See the *Reason Codes* section in the description of "DMSCOMM - Commit" on page 71 for these errors.

If your application uses high-level language statements or OS simulation in an assembler program to update data controlled by a resource that participates in CRR, be sure to empty the buffers (for example, close the files) before committing the changes. This may be required because some high-level languages and assembler programs use OS/MVS queued sequential access method (QSAM) for output files. For example, if a program uses OS/MVS QSAM with blocked records for an SFS output file, some of the most recently written output records may not be committed if the program issues a commit without first closing the QSAM file. These records will subsequently be committed when the program closes the file and either commits the work unit or allows end-of-command processing to commit the work unit. Using only CSL routines, FS macros, or the EXECIO command to write to SFS files avoids this situation.

NOCOMMIT
means do not commit changes. They will not be committed until DMSCOMM is issued or the COMMIT parameter for the same work unit ID on another routine is specified, or until successful end-of-command, where CMS implicitly commits the changes. This is a character variable with a length of 8.

wuerror
is a variable that describes a buffer where extended error information is returned. Information about some errors, such as parameter syntax errors, is not returned in the *wuerror* buffer if the error is detected before the file pool server is called.

Extended error information is returned only for operations on the SFS. Error information about operations on minidisk files is provided by the return and reason codes.

To use the *wuerror* parameter, specify a length of 12 bytes plus 272 bytes for every file pool affected by the work unit.

- Information is always returned in the first 12 bytes of the buffer. They contain the *wuerror* parameter length, the number of file pool error information groups returned, and the number of errors for which information was available from SFS. The format is shown in Table 16 on page 19.

  It is incorrect to specify a length less than 12, except zero. Specifying a length of zero has the effect of omitting the *wuerror* parameter.
- Information is returned in the rest of the buffer only if there is a warning or an error. Allocate enough space for two groups—each 136 bytes long—of error information per file pool.

  The information is formatted by the FPERROR macro, as shown in Table 17 on page 19. For an explanation of each of the fields returned—see “DMSWUERR - Work Unit Error Data Deblocker” on page 553 for routine description. Assembler programmers can also refer to the WUERROR and FPERROR assembler language macros, which are described in the *z/VM: CMS Macros and Functions Reference*.

Table 16. Format of Extended Error Information

Field Name	Field Type	Description
length	INT	Length of the *wuerror* parameter
number_returned	INT	Number of file pool error information groups returned
total_errors	INT	Total number of file pool error information groups available from SFS
FPERROR		One or more groups of file pool error information (see Table 17 on page 19)

Table 17. Format of File Pool Error Information Group

Field Name	Field Type	Description
filepoolid	CHAR(8)	File pool ID
********	CHAR(8)	Reserved
workunitid	INT	Work unit ID
error_reascode	INT	Error reason code
********	INT	Reserved
********	INT	Reserved
prev_retcode	INT	Return code
warning1...warning16	ARRAY of INT	Warning reason code (space for 16 allowed)
userid_index	INT	User ID index
level_sub1	CHAR(4)	Reserved
level_sub2	CHAR(4)	Reserved
error_reascode_info	CHAR(4)	Reason code augmentation field
failing_filepoolid	CHAR(8)	File pool ID of failing resource
********	CHAR(12)	Reserved

Notation Used in Parameter Descriptions

In this book, the description of each parameter for a CSL routine begins with the three-part notation:

(*usage, type, length*)

In this notation:

usage
is one of the following, indicating how the variable is used by the called routine:

input
means you must supply a value for the parameter in the call.

output
means the routine returns a value in the parameter when the call is finished.

input/output
means the same parameter is used to supply a value to the routine and return a value from the routine.

type
is one of the following, indicating the type of data the parameter contains:

INT
means signed binary integer.

CHAR
means character string.

length
is the length of the variable, specified as one of the following:

- A number (such as **8**), a choice of two numbers (such as **0 or 9**), or a range of numbers (such as **1-8**), indicating the number of bytes or characters (depending on the data type)

  Note: When a range is indicated for an output variable and you do not know the length of the value to be returned, you should set the variable to the maximum length to ensure that it will hold the complete returned value. Otherwise, if the returned value does not fit, you will receive an error indication. In that case, you will have to increase the length of the variable and call the routine again.
- The name of another parameter variable (such as ***length1***) that specifies the number of bytes or characters.

Using the Online HELP Facility

You can receive online information about the CSL routines described in this book using the z/VM HELP Facility. For example, to display a menu of CSL routines, enter:

```
help routine menu
```

To display information about a specific CSL routine (DMSCATTR for example), enter:

```
help routine dmscattr
```

To display information about a message (DMS001E for example), enter one of the following commands:

```
help dms001e
help msg dms001e
```

For more information about using the HELP Facility, see the *z/VM: CMS User's Guide*. To display the main HELP Task Menu, enter:

```
help
```

For more information about the HELP command, see the *z/VM: CMS Commands and Utilities Reference* or enter:

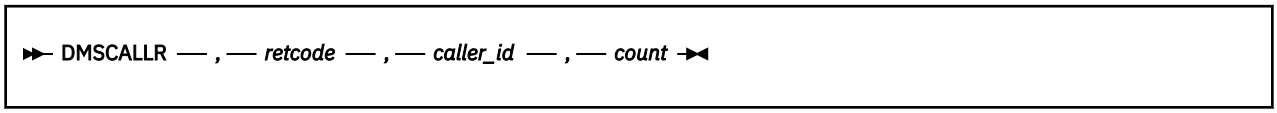
```
help cms help
```

Chapter 2. Callable Service Descriptions

This chapter describes each of the CMS callable services. The services are arranged in alphabetical order.

If you are unfamiliar with the conventions used to describe the system calls, refer to Chapter 1, “Invoking Callable Services,” on page 1.

DMSCALLR - Get Caller Identification



Call Format

The format for calling a CSL routine is language dependent. DMSCALLR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCALLR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCALLR routine to examine the command call chain.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCALLR:

Code
Meaning

0
Information about the calling command has been copied into *caller_id*.

8
The value of *count* exceeds the number of commands in the calling sequence.

caller_id
(output, CHAR, 16) is a variable to contain the name of the CMS command that is *count* commands back in the calling sequence. The *caller_id* is the real command name, after abbreviation and synonym processing has been performed.

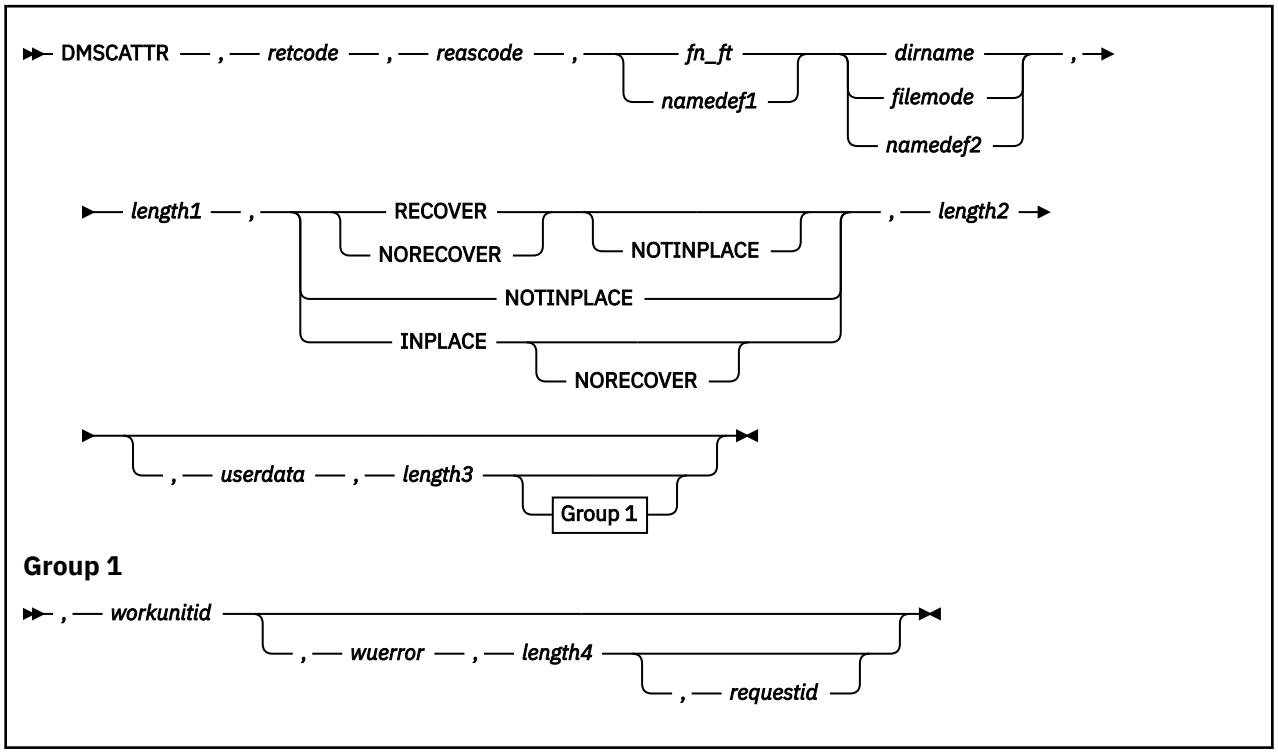
The first two tokens of the tokenized parameter list are copied into *caller_id*. Execs will have EXEC as the first token. The CMS subcommand environment will have CMS as the only token.

count
(input, INT, 4) is a variable that identifies how far back in the calling sequence for which a name is needed. A value of zero returns the name of the most recently issued command.

Usage Notes

1. When called from REXX with a value of 0 for *count*, DMSCALLR always returns the command name RXCSL.
2. To determine if your program was called from within XEDIT, you must look at each command name until you find one that is not EXEC or CMS. If the command name is XEDIT, then the program was called from an exec or XEDIT macro.

DMSCATTR - Change Attributes



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSCATTR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCATTR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCATTR routine to modify the recoverability and overwrite attributes of an SFS file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCATTR.

reascode
(output, INT, 4) is a variable for the reason code from DMSCATTR.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and the file type of the file for which the overwrite or recoverability attributes will be modified.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the directory name.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If a one character namedef is used in this field, it is treated as a namedef rather than as a file mode letter. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname* or *filemode*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, plus *dirname* or *namedef2*). See "Compound Variables" on page 15 for coding details.

RECOVER
(input, CHAR, 7) indicates that all uncommitted changes to the file are discarded when a work unit is rolled back. The rollback can be initiated by the application, or it can be caused by abend processing or system failure.

NORECOVER
(input, CHAR, 9) indicates that changes to the file are not rolled back when the application initiates a rollback. In most cases, the updates are committed.

In implicit rollbacks, changes are committed if possible; in particular, cached updates can be discarded.

NOTINPLACE
(input, CHAR, 10) causes changes to the file to be shadowed so readers see a consistent version of the file from Open to Close.

INPLACE
(input, CHAR, 7) causes updates to be made in place where possible. This can reduce DASD utilization and enables readers to see file updates by a concurrent writer. Users that have an INPLACE file open for read must reopen the file to see extensions (new records or blocks) that have been written and committed to the file.

length2
(input, INT, 4) is a variable for specifying the length, in bytes, of the preceding compound character parameter (RECOVER or NORECOVER, NOTINPLACE or INPLACE). See "Compound Variables" on page 15 for coding details.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). Specifying a value of 0 has the effect of omitting the *userdata* parameter.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended error information from CMS. See *wuerror* under "Common Parameters" on page 15 for more information. If it is specified, it must be followed by a length field.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Usage Notes

1. If either the overwrite or the recoverability attribute is not specified, it will not be modified by the operation, even if the result is the incompatible combination of RECOVER and INPLACE. In such a case, an error will be generated.
2. At least one of the keyword parameters must be specified on the call to DMSCATTR.
3. DMSCATTR is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it finishes, DMSCATTR causes a noncoordinated commit to be done for the work in the affected file pool.
4. Write authority to the file and the directory is required in order to change the attributes of a file.
5. The attributes of a file cannot be changed if the file:
   - Is locked for SHARE by you
   - Is locked for SHARE, EXCLUSIVE or UPDATE by another user
   - Is open (for read or write)
   - Has been modified and the changes have not yet been committed
6. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file or directory from which you are deleting a lock.
7. The file ID specified must refer to either a base file or an alias.
8. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
9. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCATTR, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCATTR. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

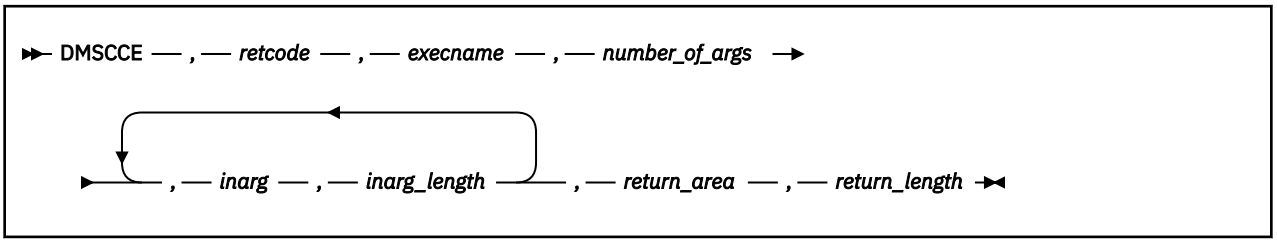
DMSCATTR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	78106	The file already has the specified attributes.
ERROR	30500	The combination of attributes RECOVER and INPLACE is not supported.

Severity	Reason Code	Description
ERROR	44000	The file does not exist or you do not have the authority to change its attributes.
ERROR	63700	The specified file resides in a directory control directory that is accessed read-only.
ERROR	64300	The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90250	The file name and file type (or *namedef*) are required but were not specified.
ERROR	90300	Incorrect parameter in CSL parameter list.
ERROR	90320	Conflicting parameters in CSL parameter list.
ERROR	90330	Duplicate parameters in CSL parameter list.
ERROR	90350	Incorrect number of blank-delimited tokens in the file ID or dirname parameter. There must be at least one and not more than four tokens in the string.
ERROR	90380	Missing parameter in CSL parameter list. Recoverability and overwrite attributes must be specified.
ERROR	90410	Incorrect parameter length specified.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the file ID parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the file ID parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or the file type part of the file ID parameter.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is incorrect.
ERROR	90505	The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the file ID or dirname parameter is no longer than 16 characters.
ERROR	90530	The namedef part of the file ID or dirname parameter does not exist or was used incorrectly. For example, a namedef that was created for a dirname was used where a file name/file type namedef was expected.
ERROR	90540	Incorrect work unit ID specified.
ERROR	90590	There is no default file pool currently defined, and the file pool ID was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.

Severity	Reason Code	Description
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

DMSCCE - Call a REXX Exec



Call Format

The format for calling a CSL routine is language dependent. DMSCCE is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCCE
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCCE" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCCE routine to invoke a REXX exec from an application program.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCCE.

execname
(input, CHAR, 8) is a variable for specifying the name of the REXX exec being invoked. If necessary, pad the exec's name with blanks on the right out to eight characters.

number_of_args
(input, INT, 4) is a variable for specifying the number of input argument strings being passed to the REXX exec. Up to 10 input character strings are allowed on a call. (See Usage Note "3" on page 31.)

inarg
(input, CHAR, *inargn_length*) is an argument string passed to the REXX exec. It must be declared with a type of character in the calling language.

inarg_length
(input, INT, 4) is the length of the preceding argument string.

return_area
(output, CHAR, *return_length*) is a buffer area to receive data from the REXX exec.

return_length
(input/output, INT, 4) on input, this is the length of *return_area*; on output, this is the length of the data returned in *return_area*. (See Usage Note "5" on page 31.) It must be specified as a variable rather than a direct value.

Usage Notes

1. This routine is useful when the application needs to invoke a CMS or CP command. The REXX exec can issue the CP or CMS command and pass the results back to the application program. To pass data back from the REXX exec, use the RETURN instruction, which is described in the *z/VM: REXX/VM Reference*.

2. One way to use DMSCCE is to issue a FILEDEF command from an application program. A REXX exec named DATADEF, supplied with VM, invokes the FILEDEF command. The following code fragment from a PL/I program shows an example of this:

```
⋮
/* Declare parameters of CALL statement */
  DCL DMSCCE    CHAR(8)       INIT('DMSCCE'),
      RETCODE   FIXED BIN(31) INIT(0),
      DATADEF   CHAR(8)       INIT('DATADEF'),
      NUMARG    FIXED BIN(31) INIT(1),
      ARG       CHAR(38) INIT('INFILE DISK
FILENAME FILETYPE A (PERM)'),
      ARGL      FIXED BIN(31) INIT(38),
      RET       CHAR(10)      INIT(' '),
      RETL      FIXED BIN(31) INIT(10);

/* Call statement to FILEDEF EXEC */
  CALL DMSCSL (DMSCCE,RETCODE,DATADEF,NUMARG,
ARG,ARGL,RET,RETL);
⋮
```

   After the application program issues the above CALL statement, the FILEDEF command is executed using the arguments supplied in the "ARG" parameter.

3. Although you cannot specify more than ten arguments on a call to DMSCCE, an argument string can represent several variables. For example, you could pass 'var1 var2 var3 var4 var5' as a single argument string, and this single string can be parsed into five separate variables.

4. The technique for issuing a CMS command in usage note “2” on page 31 can be generalized with this exec:

```
/*CMD_CALL   */
parse arg cmdname  cmdargs
address COMMAND cmdname cmdargs
return rc
```

   To use it, call DMSCCE and specify "CMD_CALL" and the CMS command and its operands as parameters.

5. If the information returned from the REXX exec is longer than the length of *return_area*, the data area is truncated and a return code of 200 is issued.

6. This routine cannot be called by a REXX exec.

Return Codes

Note that for the last two return codes listed, the *nn* designates the relative position of the parameter: *retcode* is always parameter number 01, the next parameter is number 02, and so on.

Code
Meaning

0
Normal completion.

20
Invalid CMS character in *execname*.

28
The REXX exec specified on the call does not exist.

112
The number of parameters passed on the call was incorrect.

118
The parameter list passed to the routine was not in a valid format.

123
The number of arguments passed to the REXX exec exceeded the number specified in *number_of_args*.

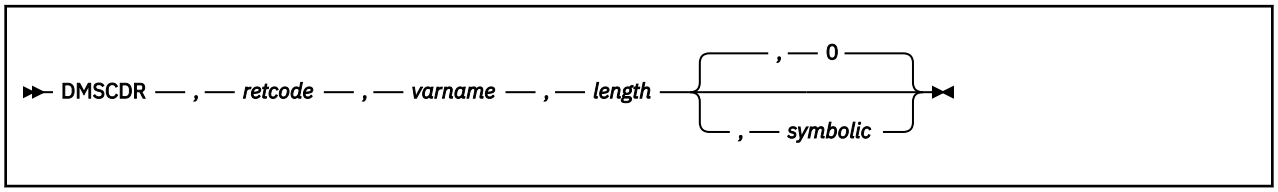
200
The data returned in *return_area* has been truncated. (The *return_length* variable contains the length of the data before it was truncated.)

10*nn*
The data type for parameter *nn* is incorrect.

20*nn*
The length for parameter *nn* is incorrect.

nnnn
Any return code that can be passed back from REXX itself or from the REXX exec that is being invoked.

DMSCDR - Drop a REXX Variable



Call Format

The format for calling a CSL routine is language dependent. DMSCDR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCDR
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCDR" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCDR routine to cause a calling REXX exec to drop a single REXX variable or a group of REXX variables having the same stem. Dropping variables restores them to their original, uninitialized state. DMSCDR must be called from a program written in a language other than REXX.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCDR.

varname
(input, CHAR, *varname_length*) is a variable for specifying the name of the REXX variable to drop. If the REXX variable name ends with a period (.), all REXX variables with that stem are dropped.

length
(input, INT, 4) is a variable for specifying the length of the name of the variable specified in *varname*.

symbolic
(input, INT, 4) is a variable for specifying a flag field. If *symbolic* is equal to zero, REXX searches for the contents of *varname* directly in its variable lists; if *symbolic* is not equal to zero, REXX does normal uppercase translation and substitution for compound variables. This field is optional and defaults to zero, meaning REXX searches for the variable name directly.

(Refer to the *z/VM: REXX/VM User's Guide* for more information about REXX translation and substitution.)

Usage Notes

1. This routine provides the equivalent of the REXX instruction DROP. For more information on the DROP instruction, see the *z/VM: REXX/VM Reference*.
2. When DMSCDR is called from a REXX program, it searches the symbol table of only that program. If the variable is not defined in the program, DMSCDR returns an error code. However, if the variable is defined, DMSCDR drops the variable and does not indicate the error.

Return Codes

Note that for the last two return codes listed, the *nn* designates the relative position of the parameter: *retcode* is always parameter number 01, the next parameter is number 02, and so on.

Code
Meaning

0
Normal completion.

112
The number of parameters passed on the call was incorrect.

201
REXX is working and cannot share variables now.

202
REXX is not active.

204
Variable name format is invalid.

207
Unsupported function. (This return code is issued if DMSCDR is called from within an EXEC2 environment.)

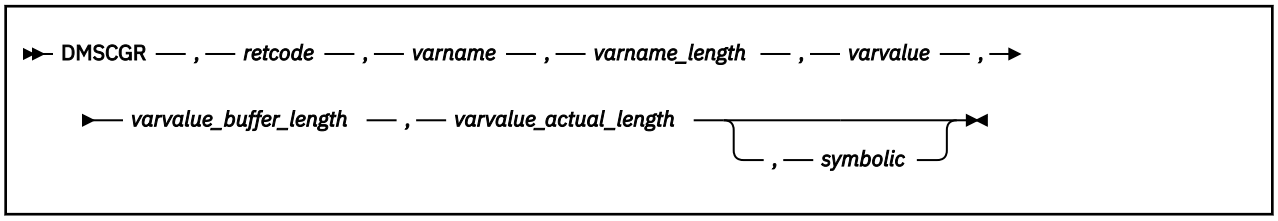
208
Insufficient storage.

209
Storage failure (error in CMSSTOR or SUBPOOL macro).

10*nn*
The data type for parameter *nn* is incorrect.

20*nn*
The length for parameter *nn* is incorrect.

DMSCGR - Get a REXX Variable



Call Format

The format for calling a CSL routine is language dependent. DMSCGR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCGR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCGR routine to retrieve a REXX variable: REXX program A calls program B, which can use DMSCGR to retrieve a variable from program A. Program B must not itself be a REXX program.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCGR.

varname
(input, CHAR, *varname_length*) is a variable for specifying the fully-qualified name of the variable whose value you want. (See Usage Note "1" on page 36.)

varname_length
(input, INT, 4) is a variable for specifying the length of *varname*.

varvalue
(output, CHAR, *varvalue_buffer_length*) contains the name of a buffer used to store the value of the variable.

varvalue_buffer_length
(input, INT, 4) is a variable for specifying the length of the *varvalue* parameter.

varvalue_actual_length
(output, INT, 4) is a variable for returning the length of the data that is assigned to *varname* and returned in *varvalue*.

If the buffer is too long, the unused part of the buffer is unchanged. If the buffer is too short, the buffer is filled and a return code of 200 is set.

symbolic
(input, INT, 4) is a variable for specifying a flag field. If *symbolic* is equal to zero, REXX searches for the contents of *varname* directly in its variable lists; if *symbolic* is not equal to zero, REXX does normal uppercase translation and substitution for compound variables. This field is optional and defaults to zero, meaning that REXX searches for the variable name directly.

For more information about REXX translation and substitution, see the *z/VM: REXX/VM User's Guide*.

Usage Notes

1. If the variable name you specify in *varname* has qualifiers, you must specify them. For example, if your variable is the name of an array, you must specify the complete name: the stem followed by a period (.) and the qualifier.
2. The DMSCGR routine is useful when a high-level language application program is called from an exec that has, for example, parsed the command line into variables. These variables are then available to the application program.
3. When DMSCGR is called from a REXX program, it searches the symbol table of only that program. If the variable is not defined in the program, DMSCGR returns an error code. However, if the variable is defined, DMSCGR returns the value and does not indicate the error. You can also retrieve REXX variables by issuing the CMS Pipelines VARS or STEM stage command from REXX and non-REXX programs (see the *z/VM: CMS Pipelines User's Guide and Reference*.

Return Codes

Note that for the last two return codes listed, the *nn* designates the relative position of the parameter: *retcode* is always parameter number 01, the next parameter is number 02, and so on.

Code
Meaning

0
Normal completion.

112
The number of parameters passed on the call was incorrect.

200
The data returned in *varvalue* has been truncated. (The *varvalue_actual_length* variable contains the length of the data before it was truncated.)

201
REXX is working and cannot share variables now.

202
REXX is not active.

203
Variable was not found.

204
Variable name format is invalid.

205
Variable name is too long. (This return code is issued from EXEC2 only.)

207
symbolic parameter not supported. (This return code is issued from EXEC2 only.)

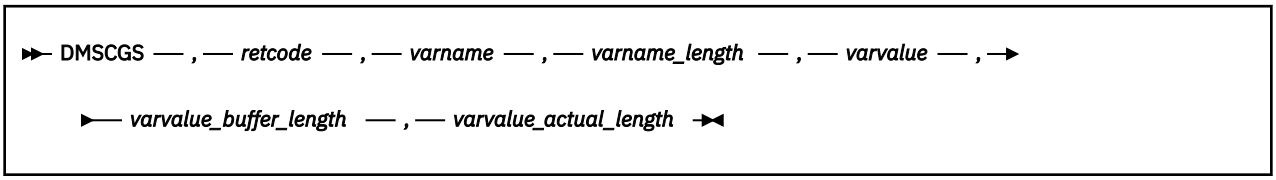
208
Insufficient storage.

209
Storage failure (error in CMSSTOR or SUBPOOL macro).

10*nn*
The data type for parameter *nn* is incorrect.

20*nn*
The length for parameter *nn* is incorrect.

DMSCGS - Get Special REXX Values



Call Format

The format for calling a CSL routine is language dependent. DMSCGS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCGS
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCGS" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCGS routine in a program called by a REXX exec to retrieve the exec's arguments, source program, or version information. The program that calls DMSCGS cannot be written in REXX.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCGS.

varname
(input, CHAR, *varname_length*) is a variable for specifying the special REXX information you want to obtain. The value ARG, PARM, PARM.*n*, SOURCE, or VERSION can be passed directly or in a variable.

Note: Except for the PARM variable ('PARM' or 'PARM.*n*'), REXX uses only the first letter of this parameter name for comparisons. You should completely spell out the names of the REXX variables you are passing to avoid possible confusion.

varname_length
(input, INT, 4) is a variable for specifying the length of the REXX variable's name.

varvalue
(output, CHAR, *varvalue_buffer_length*) is the name of a buffer used to return the REXX variable's value.

varvalue_buffer_length
(input, INT, 4) is a variable for specifying the length of the buffer *varvalue*.

varvalue_actual_length
(output, INT, 4) contains the length of the data returned in the *varvalue* parameter. (See Usage Note "2" on page 37.)

Usage Notes

1. This routine provides the equivalent of the REXX statements PARSE ARG, PARSE SOURCE, and PARSE VERSION. For a description of the PARSE instruction, see the *z/VM: REXX/VM Reference*.
2. If the length of the returned data (*varvalue_actual_length*) is shorter than the buffer space reserved (*varvalue_buffer_length*), the unused part of the buffer is unchanged.

   If the length of the returned data (*varvalue_actual_length*) is longer than the buffer space reserved (*varvalue_buffer_length*), the buffer is filled and a return code of 200 is set.

3. When DMSCGS is called from a REXX program, it returns values from that program and does not indicate the error.

Return Codes

Note that for the last two return codes listed, the *nn* designates the relative position of the parameter: *retcode* is always parameter number 01, the next parameter is number 02, and so on.

Code
Meaning

0
Normal completion.

112
The number of parameters passed on the call was incorrect.

200
The data returned in *varvalue* has been truncated. (The *varvalue_actual_length* variable contains the length of the data before it was truncated.)

201
REXX is working and cannot share variables now.

202
REXX is not active.

207
Variable name is not one of the three supported.

(This return code is also issued if DMSCGS is called from within an EXEC2 environment.)

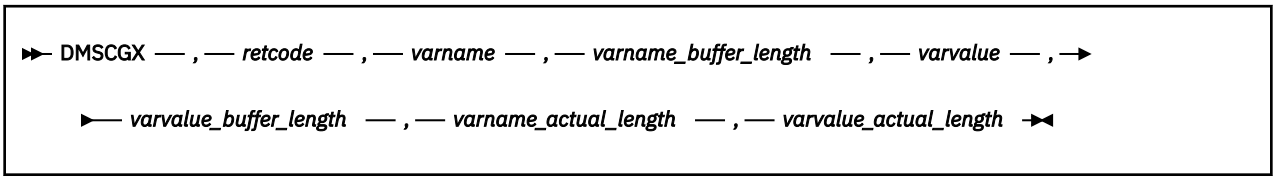
208
Insufficient storage.

209
Storage failure (error in CMSSTOR or SUBPOOL macro).

10*nn*
The data type for parameter *nn* is incorrect.

20*nn*
The length for parameter *nn* is incorrect.

DMSCGX - Get the Next REXX Variable



Call Format

The format for calling a CSL routine is language dependent. DMSCGX is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCGX
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCGX" must be padded with blanks to eight characters.

For more information and examples of the call formats, see “Calling VMLIB CSL Routines” on page 2.

Purpose

Use the DMSCGX routine in a program called from a REXX program to retrieve the name and value of a variable known to the calling exec. The program that calls DMSCGX cannot be written in REXX.

By putting a call to this routine into a loop, you can retrieve names and values of all variables known to a REXX program.

Parameters

retcode
(output, INT, 4) is a variable to hold the return code from DMSCGX.

varname
(output, CHAR, *varname_buffer_length*) is a variable used to return the fully qualified name of the REXX variable.

Note: If the variable name returned in *varname* has qualifiers, they are specified. For example, if the variable is the name of an array, the complete name—the stem followed by a period (`.`), and then the qualifier—is returned.

varname_buffer_length
(input, INT, 4) is a variable for specifying the length of the *varname* parameter.

varvalue
(output, CHAR, *varvalue_buffer_length*) is a variable for specifying the name of a buffer used to store the variable's value.

varvalue_buffer_length
(input, INT, 4) is a variable for specifying the length of the buffer *varvalue* reserved by the program.

varname_actual_length
(output, INT, 4) is a variable for returning the actual length of the REXX variable's name.

varvalue_actual_length
(output, INT, 4) is a variable for returning the actual length of the REXX variable's value.

Usage Notes

1. Each invocation of DMSCGX gets a single REXX variable, but subsequent invocations get the next REXX variable in storage. So, by invoking DMSCGX repeatedly, you can get all REXX variables. The variables are retrieved from REXX storage in an unspecified order.

2. When all the variables have been retrieved, a return code of 206 is set. The subsequent call to DMSCGX will start back at the first REXX variable.
3. The search also starts over from the beginning if a supervisor call (SVC) is issued, or if any of the following CSL routines is called:
   - DMSCCE (Call a REXX EXEC)
   - DMSCGR (Get a REXX Variable)
   - DMSCGS (Get A Special Variable)
   - DMSCSR (Set A REXX Variable)
   - DMSCDR (Drop A REXX Variable)
4. If the returned length of the variable's name or value is shorter than the buffer space reserved, the unused part of the buffer is unchanged. If the length of the variable's name or value is longer than the buffer space reserved, the buffer is filled and a return code of 200 is set.
5. When DMSCGX is called from a REXX program, it returns variables from that program and does not indicate an error. You can retrieve REXX variables from a calling REXX program by issuing the CMS Pipelines VARLOAD stage command from REXX and non-REXX programs (see the *z/VM: CMS Pipelines User's Guide and Reference*.

Return Codes

Note that for the last two return codes listed, the *nn* designates the relative position of the parameter: *retcode* is always parameter number 01, the next parameter is number 02, and so on.

Code
Meaning

0
Normal completion.

112
The number of parameters passed on the call was incorrect.

200
The data returned in *varvalue* has been truncated. (The *varvalue_actual_length* variable contains the length of the data before it was truncated.)

201
REXX is working and cannot share variables now.

202
REXX is not active.

206
All variables have been processed.

207
Unsupported function. (This return code is issued if DMSCGX is called from within an EXEC2 environment.)

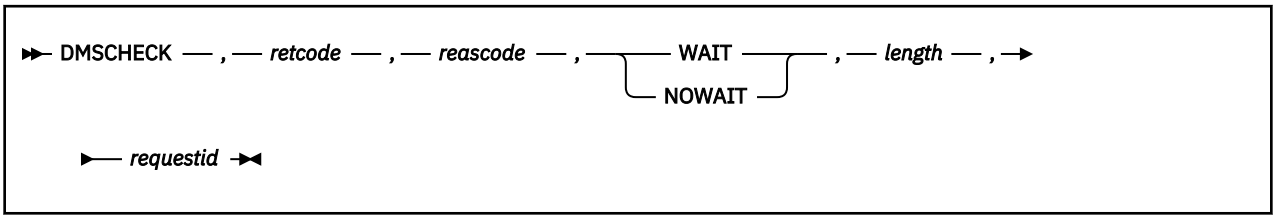
208
Insufficient storage.

209
Storage failure (error in CMSSTOR or SUBPOOL macro).

10*nn*
The data type for parameter *nn* is incorrect.

20*nn*
The length for parameter *nn* is incorrect.

DMSCHECK - Check



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSCHECK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCHECK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCHECK routine to determine if an asynchronous request has been completed.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCHECK.

reascode
(output, INT, 4) is a variable for the reason code from DMSCHECK.

WAIT
(input, CHAR, 4) indicates that control is not to be returned to the caller until the request identified by *requestid* has been completed. This parameter can be specified as a character literal or it can be passed in a variable.

NOWAIT
(input, CHAR, 6) indicates that if the specified request has not yet been completed, control is returned to the caller with a warning return code and reason code. If the request has already been completed, control is returned to the caller with a return code of 0. This parameter can be specified as a character literal or it can be passed in a variable.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (WAIT or NOWAIT).

requestid
(input/output, INT, 4) specifies the request to be checked. A request ID of 0 means check any outstanding request. If a specific request has been completed, the request is returned to the caller. If the request ID is not 0, it must be the same one that was returned from the original request.

Usage Notes

1. If the return and reason codes passed back by DMSCHECK indicate that the request has been completed, you can access the variables you passed on that asynchronous request. The return and

reason codes associated with that request indicate the success of the operation. Be sure to use different variables for the return code values on your calls to DMSCHECK and to the CSL routine being called asynchronously. If you specified 0 for *requestid*, the *requestid* parameter returned by DMSCHECK will indicate which asynchronous request has been completed.

2. Issuing a DMSCHECK on a request that was not accepted results in an error. In this case, the proper return and reason codes are returned indicating that the request was not accepted.
3. If the work unit for outstanding *non-atomic* work is rolled back, the communication path for the request is severed. In this situation, a call to DMSCHECK fails, because the specified request ID does not match any active request.

   If the work unit for outstanding *atomic* work is rolled back, the communication path for the request is not severed, since atomic requests are not affected by a rollback. Your application can continue to use DMSCHECK to check for completion.
4. If a program ends without completing a check for an outstanding asynchronous request, the work unit containing that request is rolled back.
5. When the WAIT parameter is specified, your virtual machine enters a *wait state*: No work can be done by the virtual machine until the request has been completed. If the *requestid* parameter indicates a particular request, no work is done until that request is complete. If 0 is specified for *requestid*, your virtual machine remains in a wait state only until the next outstanding request is complete.
6. When NOWAIT is specified, the application must be running in an interrupt-enabled environment.
7. For information on checking for the completion of asynchronous SFS requests issued from multitasking applications, see the *z/VM: CMS Application Development Guide*.

Return Codes and Reason Codes

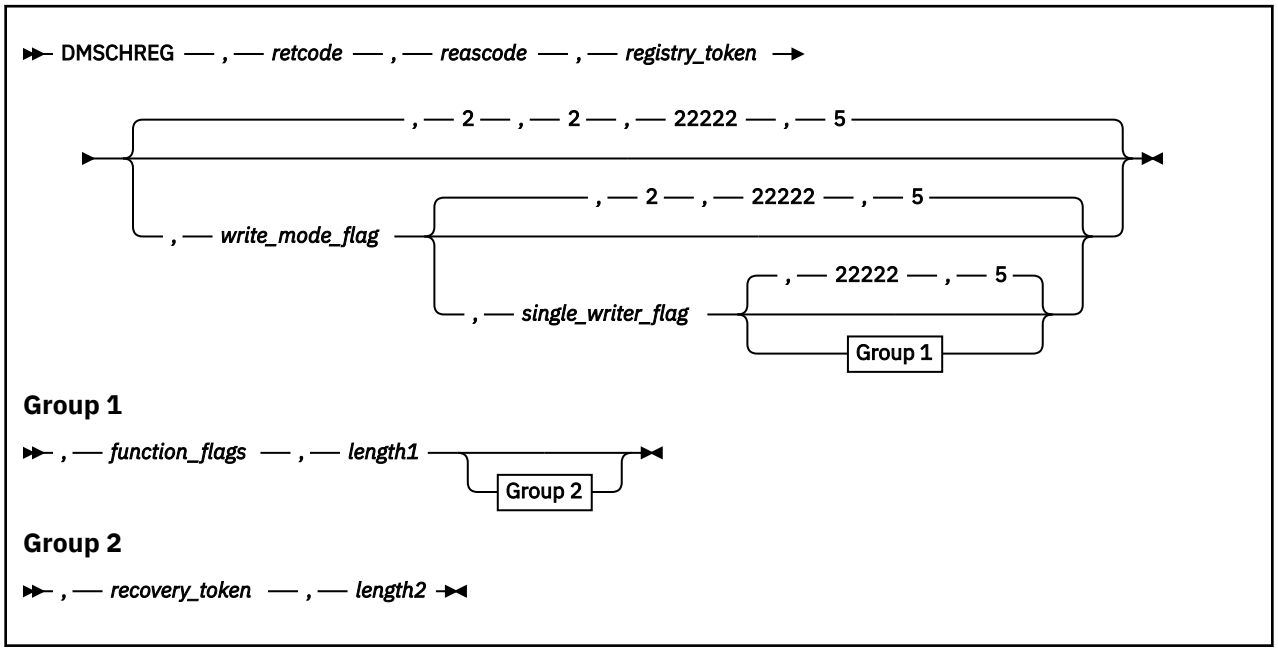
For lists of the possible return codes from DMSCHECK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCHECK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSCHECK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	90222	The asynchronous requests have not been completed.
ERROR	90216	The specified request ID does not match any active request.
ERROR	90477	Invalid wait option specified, must be WAIT or NOWAIT
ERROR	90555	0 was specified for the request ID, but there are no active asynchronous requests.

DMSCHREG - CRR Change Registration



Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSCHREG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCHREG
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCHREG routine to change certain registry values for a resource registered with the CRR synchronization point manager (SPM). For example, the resource adapter should call this routine to change the function flags when the resource is not *in work* and the resource adapter *temporarily* does not want to participate in sync point activities.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCHREG.

reascode
(output, INT, 4) is a variable for the reason code from DMSCHREG.

registry_token
(input, CHAR, 8) is a variable that identifies the registration being changed. This value must have been previously returned by a successful call to DMSREG, or unpredictable results could occur.

write_mode_flag
(input, CHAR, 1) is a variable for indicating whether this resource should be treated as if there are updates to be committed or backed out:

1
ON. The resource should be treated as write-mode. A commit of the resource would leave it in a different state than a backout.

0
OFF. The resource may be treated as read-only. No recovery of the resource is needed in case of a failure during sync point processing, because from a recovery point of view a backout of the resource is equivalent to a commit. Therefore, the SPM and the CRR recovery server do not do any logging for this resource.

2
CURRENT. Keep the current setting. This is the default.

single_writer_flag
(input, CHAR, 1) is a variable for indicating, when the write mode flag is set on, whether this resource is the only write-mode resource permitted for the work unit:

1
ON. This is the only write-mode resource permitted for the work unit. This is a compatibility interface provided to resource adapters communicating with resource managers that do not support two-phase commit.

0
OFF. Other resources can be in write mode for the work unit at the same time as this resource.

2
CURRENT. Keep the current setting. This is the default.

function_flags
(input, CHAR, 0 or 5) is a field in which the characters are positional values that indicate (left to right) whether the SPM should call an exit to this resource adapter for the following sync point functions:

- Precoordination
- Coordination
- Postcoordination
- End of work unit
- Backout required

Each character can have a value of 1, 0, or 2:

1
ON. Call the resource adapter's exit routine for this function. For all functions except backout-required, the SPM calls the *exit_name* routine specified in the registration (DMSREG). For the backout-required function, the SPM calls the *backout_exit_name* routine, if specified in DMSREG. If a *backout_exit_name* routine was not specified in DMSREG, the SPM calls the *exit_name* routine.

0
OFF. Do not call the resource adapter's exit routine for this function.

2
CURRENT. Keep the current setting for this function. If this parameter is not specified, this is the default for all positions.

length1
(input, INT, 4) is a variable for specifying the length of the *function_flags* field.

recovery_token
(input, CHAR, 0–8) is a variable for a value defined by the resource manager that is used to match up processes during resynchronization recovery if the logical unit of work identifier (LUWID) does not contain enough information to uniquely identify the indoubt process.

length2
(input, INT, 4) is a variable for specifying the length of the *recovery_token* variable.

Usage Notes

1. For guidance information on using the DMSCHREG routine in the context of getting a resource manager to participate in CRR, see the *z/VM: CMS Application Development Guide*.
2. Parameters may not have leading blanks.
3. Changes made to registry values while the coordination stage of a sync point is in progress will not take effect until the coordination stage completes.
4. If the simple commit flag was set off in the registration (DMSREG), CRR uses the two-phase commit protocol, including logging, regardless of the setting of the single writer flag.
5. A resource adapter should always leave the end-of-work-unit flag set on in the *function_flags* parameter so the resource adapter can be notified to clean up and unregister when the work unit completes. Clean-up might involve such things as releasing storage and severing paths. Resource adapters must unregister as part of their end-of-work-unit processing.
6. If the resource adapter calls DMSCHREG and gets a return code stating the registration change failed because the CRR recovery server is not available, the resource adapter must not allow any updates to be made to the resource that require coordination through CRR.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCHREG, see Appendix D, “Return Codes,” on page 597.

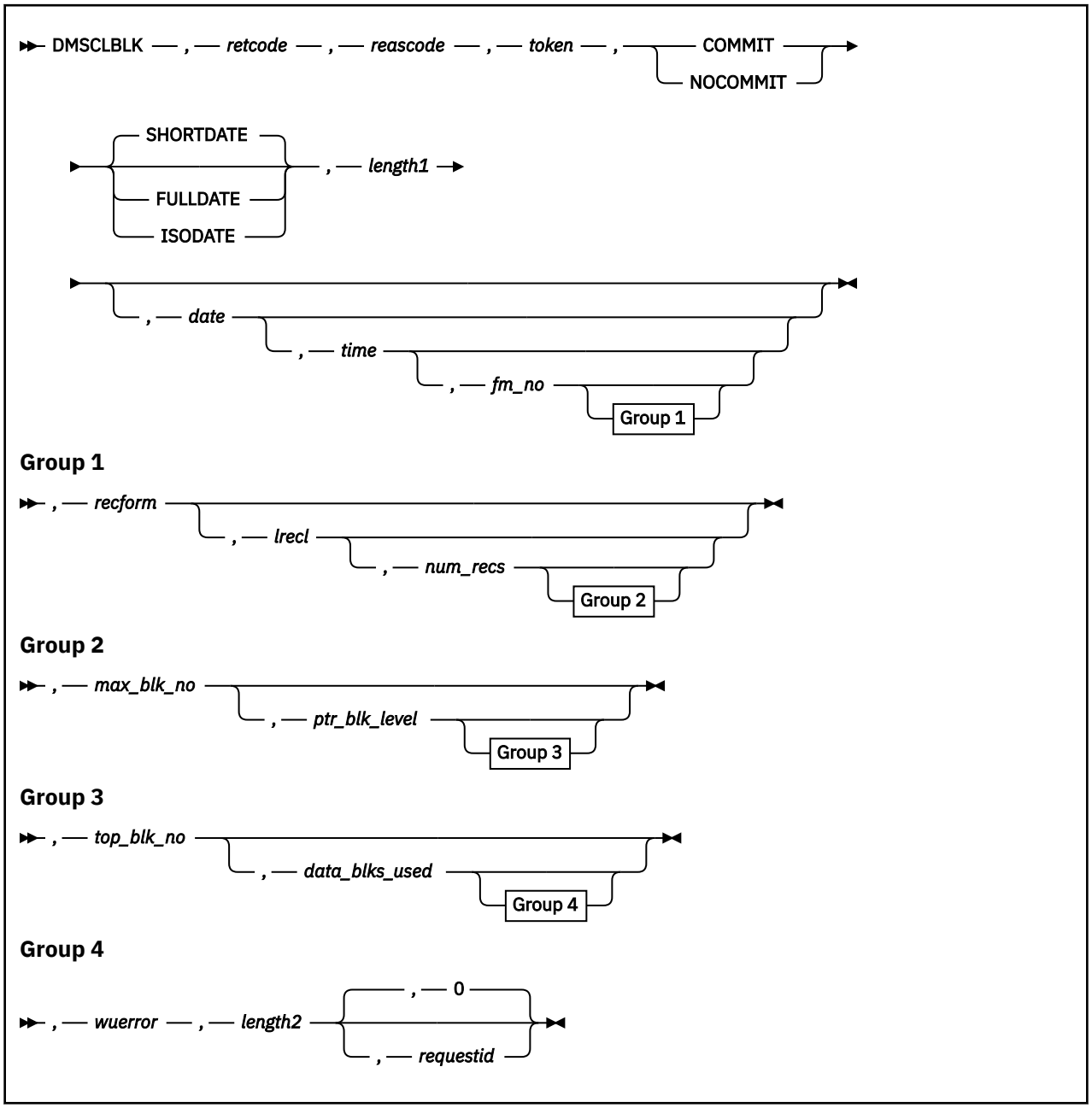
The following table lists the special reason codes returned by DMSCHREG. WARNING means the request was processed, but a warning condition was encountered; ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8.

DMSCHREG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Meaning
WARNING	79053	The CRR recovery server is not available, so the resource registration has been changed to single writer.
ERROR	55000	Insufficient virtual storage in the CRR recovery server.
ERROR	75000	The service levels of the CRR recovery server and the user machine are not compatible.
ERROR	78003	Incorrect *registry_token* parameter.
ERROR	78012	The *recovery_token* parameter is not 0–8 bytes in length.
ERROR	78015	The *write_mode_flag* parameter is not 0, 1, or 2.
ERROR	78016	The *single_writer_flag* parameter is not 0, 1, or 2.
ERROR	78017	The *function_flags* parameter is not five characters long, or contains a character other than 0, 1, or 2.
ERROR	79054	The CRR recovery server is needed to do a two-phase commit of the work unit, but the server is not available.
ERROR	79055	Write mode was set on, but a different resource is already in write mode for the work unit with single writer set on.
ERROR	79056	Write mode and single writer were set on, but a different resource is already in write mode for the work unit.

Severity	Reason Code	Meaning
ERROR	79058	Change Registration failed because a sync point is in progress for the work unit.
ERROR	95200	System error. Further attempts to access the CRR recovery server will be rejected.
ERROR	97280	Your attempt exceeds the number of APPC/VM connections allowed for the CRR recovery server.

DMSCLBLK - Close Blocks



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSCLBLK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCLBLK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCLBLK routine to end processing of a file that was opened previously using the DMSOPBLK (Open Blocks) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCLBLK.

reascode
(output, INT, 4) is a variable for the reason code from DMSCLBLK.

token
(input, CHAR, 8) is a variable returned to the caller on Open Blocks (DMSOPBLK).

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit. That is, all changes made during a work unit, from either the start of the work unit or from the last COMMIT, are kept.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length1
(input, INT, 4) is a signed variable containing the length of the preceding character parameter (COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for details on coding compound variables.

date
(input, CHAR, 8 or 10) is the *date* attribute that will be saved with the file. The *date* attribute is the date that the file was last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *date* variable, or leave blanks, the system generates the date. The value is in local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the *time* attribute that will be saved with the file. The *time* attribute is the time at which the file was last updated. This is a variable in the form *hh:mm:ss*, where *hh* is the hour of the day in 24-hour notation, *mm* is the minutes, and *ss* is the seconds. If you omit this field or set it to 8 blanks, the system determines the time.

If you specify a time, a colon (:) must be used as the separator character and 2 digits must be specified in each position. For example:

```
12:01:09
```

fm_no
(input, CHAR, 1) is a variable for specifying the file mode number of the file you are closing.

recform
(input, CHAR, 1) is a variable for specifying whether the file being closed consists of fixed-length or variable-length records. Acceptable values are:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

lrecl
(input, INT, 4) is a variable for specifying the logical record length of the file being closed.

num_recs
(input, INT, 4) is a variable for specifying the number of records in the file.

max_blk_no
(input, INT, 4) is a variable for specifying the greatest block number for the file being closed.

ptr_blk_level
(input, INT, 4) is a variable for specifying the number of pointer block levels used in the file being closed.

top_blk_no
(input, INT, 4) is a variable for specifying the block number of the first block of the file being closed.

data_blks_used
(input, INT, 4) is a variable for specifying the number of data blocks used in the file being closed.

wuerror
(output, CHAR, *length2*) is a variable containing the information necessary to have CMS return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a signed variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. If the Open Blocks intent was for:

- NEW, WRITE, REPLACE, or CREATEMIG of an SFS file
- REPLACE of a BFS file

the values of the parameters listed below must match the values for the file being closed.

The values of these parameters are returned by DMSOPBLK when the intent is for READ, NEW, WRITE, REPLACE, or CREATEMIG. If you change the file, you have to update the parameters before you close the file.

If the Open Blocks intent was for REPLACE of a BFS file, all of the following parameters must be specified with the indicated values:

Parameter
Required Value

fm_no
1

recform
F

lrecl
1

num_recs
The number of bytes in the file

max_blk_no
data_blks_used + *ptr_blk_level*

ptr_blk_level
The number of generated pointer block levels in the file

data_blks_used
The number of 4KB data blocks in the file

top_blk_no
data_blks_used + *ptr_blk_level*

2. File mode 3 files are not erased after reading.
3. The application is responsible for closing any files that it opened with Open Blocks (DMSOPBLK). CMS will roll back any work units that have unclosed files (opened by DMSOPBLK) at normal end and abnormal end-of-command.
4. If a file or directory is open on a work unit and COMMIT is specified, the file or directory remains open and all changes are committed. A rollback does not occur.
5. The date and time attributes are updated only when the file has been created or changed:
   - You wrote to the file successfully
   - Or you are creating an empty SFS file or are replacing an SFS or BFS file with an empty file (see the ALLOWEMPTY parameter of the DMSOPBLK routine, "DMSOPBLK - Open Blocks" on page 298).

   If the file is a BFS file and the date provided is earlier than January 1, 1970, the time and date are set to 00:00:00 and January 1, 1970.
6. Opened files that have been modified through DMSWRBLK prevent a commit. However, you can commit on the close of this file, when it is the only such file open.
7. Catalogs may not be open at commit time as the result of an OPEN CATALOG request.
8. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Close Blocks operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the request to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
9. A request ID is returned if the request is to be asynchronous.

10. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
11. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
12. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
13. When *date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
14. When *date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
15. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCLBLK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCLBLK. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) routine for other possible codes.

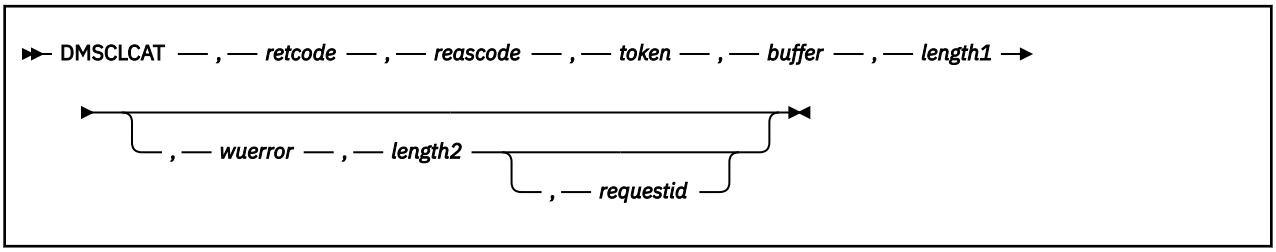
DMSCLBLK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	10050	No write was done for a new file created as a result of OPEN NEW or OPEN WRITE. Upon closing, the file no longer exists.
WARNING	10070	Open intent was REPLACE, but no write was issued. Original file is kept.
WARNING	10220	File mode other than 1 specified for a BFS file.
WARNING	51050	File space warning threshold reached or exceeded.
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
ERROR	10000	System error. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. Conflicting file attributes. Number of blocks does not match MAXBLOCK or not all blocks have been written.

Severity	Reason Code	Description
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	50500	The operation was successful but the work could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	Storage group space limit exceeded. Applicable only when COMMIT parameter is specified.
ERROR	51100	System error. No minidisks assigned to the storage group. Applicable only when COMMIT parameter is specified.
ERROR	65400	Request cannot be performed on a BFS object due to one or more of the following: • *lrecl* value other than 1 specified • *recform* value other than F specified
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back. Applicable only when COMMIT parameter is specified.
ERROR	90129	There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available. Applicable only when COMMIT parameter is specified.
ERROR	90310	Incorrect option in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect requestid specified, must be 0 or 1.
ERROR	90482	Attributes parameter was not specified for a file opened with intent of NEW, WRITE, or REPLACE.
ERROR	90492	Valid parameter must be COMMIT or NOCOMMIT
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified

Severity	Reason Code	Description
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Nonnumeric value in date specification.
ERROR	90498	Incorrect time format; must be in the form *hh:mm:ss.*
ERROR	90499	Nonnumeric value in time specification.
ERROR	95600	You have another file pool object open and specified COMMIT.
ERROR	95700	System error. No open file pool object found for the specified token.
ERROR	96500	COMMIT was specified, and there is an asynchronous request in process for the specified work unit.

DMSCLCAT - Close Catalog



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSCLCAT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCLCAT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCLCAT routine to close the object specified in a previous DMSOPCAT (Open Catalog) call.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCLCAT.

reascode
(output, INT, 4) is a variable for the reason code from DMSCLCAT.

token
(input, CHAR, 8) is a variable returned to the caller on the Open Catalog (DMSOPCAT).

buffer
(output, CHAR, *length1*) is an output buffer of at least 12 bytes which, upon completion, contains the following:

- The length of the buffer needed, including this length field (length of 4)
- The number of duplicate user IDs in the list (length of 4)
- The number of deleted user IDs in the list (length of 4)
- The list of user IDs (each user ID has a length of 8)

The list of user IDs is deleted by the system and is therefore unusable on the next request on this work unit.

If the buffer passed on input is too small, as much data as will fit is placed in the buffer and warning return and reason codes are returned. To get all of the information available you can then issue a call to DMSCPYBF (DMS Copy Buffer) with the proper buffer size by using the length given in the buffer on output. This field is only valid when the Open Catalog (DMSOPCAT) was for a storage group. While it is always specified, it is not utilized if not a storage group.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*). A length of at least 12 must be specified.

wuerror
(output, CHAR, *length2*) is an area containing the information necessary to have CMS return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable used to identify a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. A commit of the work unit is done by this routine.
2. The application is responsible for closing any catalog that it opens. At end-of-command, CMS will roll back any work units with open catalogs.
3. When writing storage group catalog information, a user ID that is to be restored may already be currently enrolled in a different storage group in the file pool. This could happen if a user was moved from this storage group to another group after the backup of the storage group occurred. If this user ID were to be restored, it would be a duplication. This duplication in two storage groups is not allowed. This user ID is not restored, but is put in a list and returned to the caller on a Close Catalog (DMSCLCAT).
4. A user may also be found in a storage group that was not there at the time of the backup. This user is deleted from the storage group, and the user ID is placed on the list of user IDs and returned to the caller on a Close Catalog (DMSCLCAT).
5. The list of user IDs is deleted by the system and is therefore unusable on the next request on this work unit.
6. A request ID is returned if the request is to be asynchronous.
7. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
8. DMSCLCAT will fail if one or more files are open for the work unit.
9. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

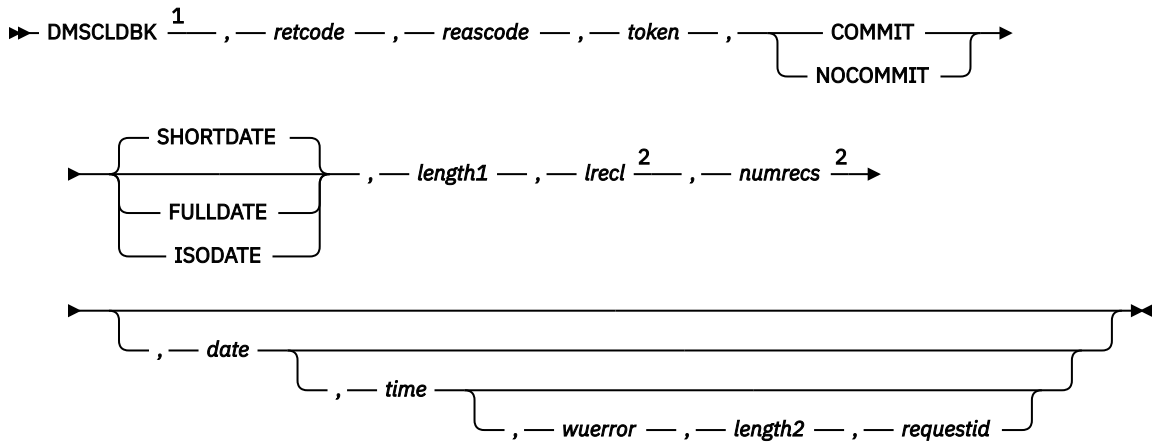
For lists of the possible return codes from DMSCLCAT, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCLCAT. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSCLCAT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	57050	You attempted to restore a user ID that is currently enrolled in other storage groups. Such user IDs are returned in a list as described in Close Catalog output. The file spaces of the listed user IDs were not restored into the storage group.
WARNING	57080	User IDs were deleted from the storage group. If a user ID was found in the storage group but not in the restore file, the file space of the user ID was deleted from the storage group. These user IDs are returned in a list as described by Close Catalog output.
WARNING	90270	Output buffer was too small to contain all of the requested output. The output has been truncated to the buffer length.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	56400	System error. The catalog is not open.
ERROR	73200	System error. For a Close Catalog request for a file space opened with intent to write, an EXCLUSIVE explicit lock on the storage group was not found.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid requestid specified, must be 0 or 1.
ERROR	90485	Invalid buffer length specified.
ERROR	95600	You have another file pool object open for the specified work unit.
ERROR	95700	System error. No open file pool object found for the specified token.
ERROR	96500	There is an asynchronous request in process for the specified work unit.
ERROR	96800	One or more files or directories are open for the specified work unit ID.

DMSCLDBK - Close Data Block



Notes:

[1] Omit the comma after the routine name when the routine is called directly.

[2] The logical-record-length and number-of-records parameters are required for a file that was opened for NEW or REPLACE. They are not required for a file that was opened for reading.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for invoking a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSCLDBK, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCLDBK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCLDBK routine to end processing of a file previously opened using the DMSOPDBK (Open Data Block) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCLDBK.

reascode
(output, INT, 4) is a variable for the reason code from DMSCLDBK.

token
(input, CHAR, 8) is a variable for uniquely identifying the file to be closed. This token is returned to the caller by DMSOPDBK.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit (including changes made to other files), from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length1
(input, INT, 4) is a variable containing the length of the preceding character parameter (COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE). See "Compound Variables" on page 15 for coding details.

lrecl
(input, INT, 4) is a variable for specifying the logical record length of the file. This parameter is required for a file that was opened for NEW or REPLACE. A warning is issued if *lrecl* does not match the length of the longest record written to a variable format file. The logical record length of the file can be determined using the Exist (DMSEXIST), Exist - File (DMSEXIFI), or Open Directory (DMSOPDIR) routine.

For a BFS file, this value must be 1.

When the opening intent was READ, *lrecl* is not required; if it is specified, it is not used but it must be correct.

numrecs
(input, INT, 4) is a variable for specifying the number of records in an SFS file or the number of bytes in a BFS file. This parameter is required for a file that was opened for NEW or REPLACE. A warning is issued if the number of records or bytes written to a variable format file is greater than *numrecs*. The number of records or bytes in the file can be determined using the Exist (DMSEXIST), EXIST - File (DMSEXIFI), or Open Directory (DMSOPDIR) routine.

When the opening intent was READ, *numrecs* is not required; if it is specified, it is not used but it must be correct.

date
(input, CHAR, 8 or 10) is the date that the file was last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *date* variable, or leave blanks, the system generates the date. The value is in local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the time at which the file was last updated. The contents of this variable must be specified in the form *hh:mm:ss*, where *hh* is the hour of the day in 24-hour notation, *mm* is the minutes, and *ss* is the seconds. To use the system-determined time, either omit this parameter or specify 8 blanks.

If you specify a time, a colon (:) must be used as the separator character and 2 digits must be specified in each position. For example:

```
12:01:09
```

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and results in an error return code.

requestid
(input, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

The processing of a given DMSCLDBK request may not require communication with the file pool server. In this case, the operation is performed synchronously regardless of the value specified in *requestid*, and the value of *requestid* is not be changed by the operation.

Usage Notes

1. If another file or a directory is open on the work unit and COMMIT is specified, the other file or the directory remains open and all changes are committed. There are, however, some situations that can prevent a commit:
   - SFS or BFS files modified using the DMSWRBLK (Write Blocks) routine remain open on the work unit.
   - SFS or BFS Catalogs (opened by the DMSOPCAT (Open Catalog) routine) remain open on the work unit.
   - SFS or BFS files opened using the DMSWRDBK (Write Data Block) remain open on the work unit.
2. An empty SFS or BFS file may be created if ALLOWEMPTY was coded on DMSOPDBK and the input number of records is equal to zero. Empty files cannot be created on a minidisk. If the input number of records specified is zero and the open intent is REPLACE for a minidisk file, the file is erased.
3. The date and time attributes are updated only when the file has been created or changed:
   - You wrote to the file successfully
   - Or you are creating an empty SFS file or are replacing an SFS or BFS file with an empty file (see the ALLOWEMPTY parameter of "DMSOPDBK - Open Data Block" on page 319).

   If the file is a BFS file and the date provided is earlier than January 1, 1970, the time and date are set to 00:00:00 and January 1, 1970.

4. The *numrecs* and *lrecl* determine the logical end of file when the file has fixed format records. If the logical end of file is greater than the last data block written, the file is extended with sparse blocks. If the logical end of file is shorter than the number of blocks written, the file is truncated. The last data block following the logical end of file consists of zeros.

   When the file has variable length records, the number of records and record length is determined by the file system (using the record length prefixes). The first record length prefix of zero determines the end of the file. If *numrecs* is smaller than the file system determined number of records, the file is truncated to the value specified by *numrecs*.
5. For a minidisk file, when the file is closed all updates to the file have been made. If COMMIT was specified, the work unit is committed. If an error occurs during an attempt to commit the work unit, the operations associated with that work unit remain uncommitted but the minidisk file is closed (the file is updated).
6. For an asynchronous request, a return code is given indicating whether the request was accepted for processing or was immediately rejected.
7. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
8. All minidisk requests are done synchronously.
9. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
10. When *date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
11. When *date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
12. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCLDBK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCLDBK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

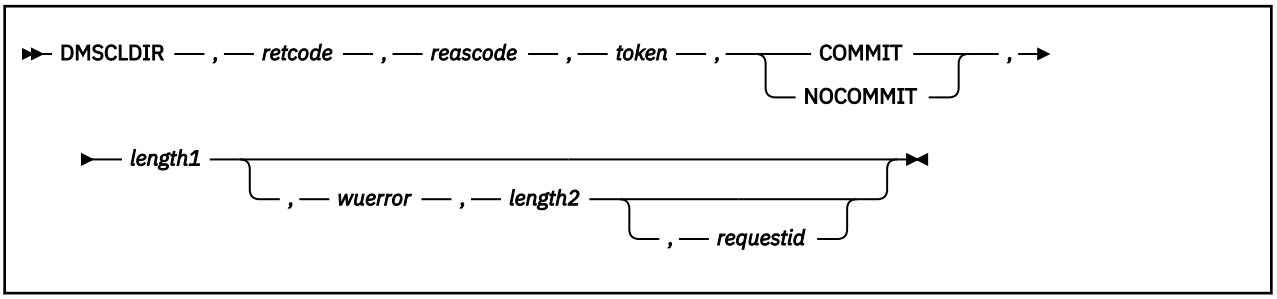
Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) routine for other possible codes.

DMSCLDBK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	10050	No file has been created. The file was empty when you tried to close it, because: • Either DMSOPDBK NEW was specified and the file was not written to • Or an empty file was not created because either ALLOWEMPTY was not specified on the OPEN, or the file pool or minidisk does not support empty files (file pool server is not at the z/VM Version Release 1.1 level or higher.)
WARNING	10070	The open intent was REPLACE and no write was issued. The original file is kept, either because DMSOPDBK was not specified with ALLOWEMPTY, or because the file pool server is not at the z/VM Version 1 Release 1.1 level or higher and does not support empty files.
WARNING	51050	File space warning threshold reached or exceeded.
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit processing.
WARNING	90617	The input file was erased. This occurs for a minidisk file when the number of records is zero. For an SFS file, this occurs when the number of records is zero and the ALLOWEMPTY parameter was not specified when the file was opened using DMSOPDBK and blocks have been written to the file.
WARNING	90691	The input value for the number of records is greater than the number of records written to a variable format file, or the input value for the logical record length does not match the length of the longest record actually written to a variable format file.
ERROR	10000	System error. Attempt to write a block but the file is not open for NEW or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	50500	Attempt to exceed the number of 4KB file blocks allowed for this user. The processing requested by this routine was performed, but has not been committed. Your application must either take remedial action (such as erase some other files in the affected file space) and then call DMSCOMM for this work unit, or call DMSROLLB to roll back this work unit.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	Storage group space limit exceeded.
ERROR	51100	System error. No minidisks assigned to the storage group.
ERROR	65400	Request cannot be performed on a BFS object because a *lrecl* value other than 1 was specified

Severity	Reason Code	Description
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90106	The input number of records was invalid. This must be a positive integer value.
ERROR	90129	There are already 2^{31} - 1 blocks in the file, therefore a new logical block is not available.
ERROR	90131	Insufficient minidisk space available.
ERROR	90300	Illegal parameter specified. The COMMIT or NOCOMMIT parameter was missing or specified incorrectly, or SHORTDATE, FULLDATE, or ISODATE was specified incorrectly, or extraneous parameters were found.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90307	Invalid *lrecl* specified. Must be a positive integer.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero *wuerror* length must be at least 12 bytes.
ERROR	90472	Invalid *requestid*. It must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Nonnumeric value in date specification.
ERROR	90498	Invalid time format; must be in the form HH:MM:SS.
ERROR	90499	Nonnumeric value in time specification.
ERROR	90690	*lrecl* and *numrecs* are required parameters when the file has been opened for NEW or REPLACE.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Data Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.
ERROR	95750	No file opened using DMSOPDBK found for the specified token.
ERROR	95777	The ERASE function issued by DMSCLDBK for a file mode number 3 file failed.

DMSCLDIR - Close Directory



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSCLDIR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCLDIR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCLDIR routine to close a directory previously opened using the DMSOPDIR (Open Directory) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCLDIR.

reascode
(output, INT, 4) is a variable for the reason code from DMSCLDIR.

token
(input, CHAR, 8) is a variable returned to the caller by DMSOPDIR. It uniquely identifies the directory you want to close.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (COMMIT or NOCOMMIT).

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and results in an error return code.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If, on return, the request ID is still 1, no server call was needed, and it will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Close Directory operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* field in one of the FPERROR entries. parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
2. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
3. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
4. Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCLDIR, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSCLDIR. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

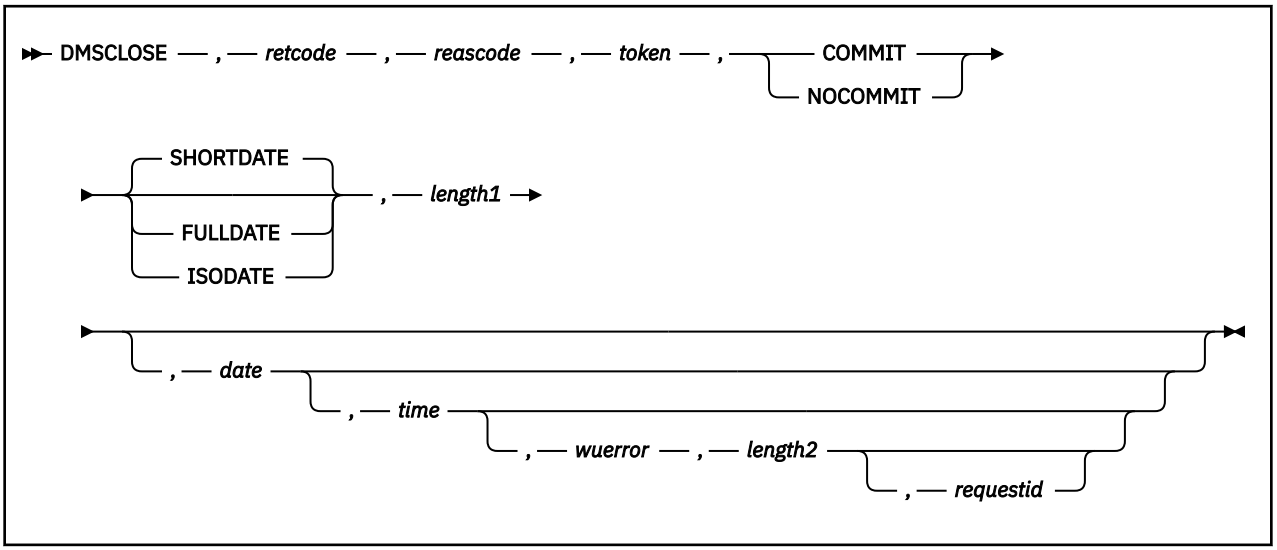
Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) routine for other possible codes.

DMSCLDIR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.

Severity	Reason Code	Description
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	60100	One or more files are open in the directory. Applicable only when the directory has the DIRCONTROL attribute and is not in a data space.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90300	Illegal parameter specified. The COMMIT/NOCOMMIT parameter was missing or specified incorrectly, or extraneous parameters were found.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSCLOSE - Close



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSCLOSE is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCLOSE
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCLOSE routine to end processing of a file previously opened using the DMSOPEN (Open) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCLOSE.

reascode
(output, INT, 4) is a variable for the reason code from DMSCLOSE.

token
(input, CHAR, 8) is a variable for uniquely identifying the file to be closed. This token was returned to the caller by DMSOPEN.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit (including changes made to other files), from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

DMSCLOSE of a BFS file causes the updates in that file to be committed even if NOCOMMIT is specified. However, this does not affect other participating resources updated on the work unit, including SFS files.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length1
(input, INT, 4) is a variable containing the length of the preceding character parameter (COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE). See "Compound Variables" on page 15 for coding details.

date
(input, CHAR, 8 or 10) is the date attribute that will be saved with the file. The date attribute is the date that the file was last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want to use a 4-digit year.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *date* variable, or leave blanks, the system generates the date. The value is in local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the *time* attribute that will be saved with the file. The *time* attribute is the time at which the file was last updated. The contents of this variable must be specified in the form *hh:mm:ss*, where *hh* is the hour of the day in 24-hour notation, *mm* is the minutes, and *ss* is the seconds. To use the system-determined time, either omit this parameter or specify eight blanks.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. For more information, see *wuerror* under "Common Parameters" on page 15.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid* which is passed on a later Check (DMSCHECK) request.

The processing of a given DMSCLOSE request may not require communication with the server. In this case, the operation is performed synchronously regardless of the value specified in *requestid*, and the value of *requestid* is not changed by the operation.

Usage Notes

1. DMSCLOSE of a BFS file causes the BFS updates to be committed whether or not COMMIT is specified. This does not affect other participating resources updated on the work unit, including SFS files. Changes made to BFS files are not protected. That is, they do not participate in CRR and are committed independently of other resources, including SFS, on the work unit.
2. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Close operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
3. An empty file is created if ALLOWEMPTY was coded on DMSOPEN and there are no records in the file at the time DMSCLOSE is issued. If a minidisk file was opened with an intent of REPLACE and no records were written, the file is erased. Empty files cannot be created on a minidisk.
4. The date and time attributes are updated only when the file has been created or changed and:
   - You wrote to the file successfully
   - Or you are creating an empty SFS file or are replacing an SFS or BFS file with an empty file (see the ALLOWEMPTY parameter of "DMSOPEN - Open" on page 340).

   If the file is a BFS file and the date provided is earlier than January 1, 1970, the time and date are set to 00:00:00 and January 1, 1970.
5. When a minidisk file is closed, all updates to the file have been made. If COMMIT was specified, the work unit will be committed. If an error occurs during the attempt to commit the work unit, the operations associated with that work unit will remain uncommitted but the minidisk file will be closed (the file is updated).
6. For an asynchronous request, the return code indicates only whether the request was accepted for processing (rc=0) or was immediately rejected; processing errors can still occur. In contrast, a return code of 0 for a synchronous request means the operation was completed successfully.
7. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
8. Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.
9. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
10. When *date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
11. When *date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.

12. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCLOSE, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSCLOSE. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

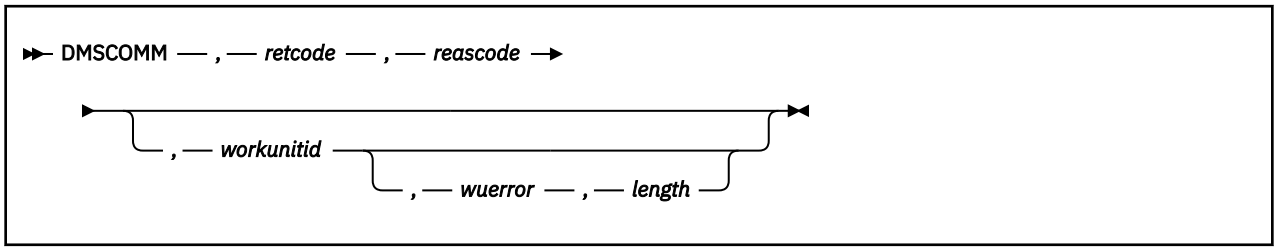
Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

DMSCLOSE can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	10050	No file has been created. The file was empty when you tried to CLOSE it, either because: • OPEN NEW was specified and the file was not written to • Or because OPEN WRITE was specified for a file that did not exist and the file was not written to. An empty file was not created either because ALLOWEMPTY was not specified on the OPEN or because the file pool or minidisk does not support empty files.
WARNING	10070	The open intent was REPLACE and no write was issued. The original SFS file is kept, either because DMSOPEN was not specified with ALLOWEMPTY, or because the file pool server does not support empty files.
WARNING	51050	File space warning threshold reached or exceeded.
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit processing.
WARNING	55777	The ERASE function issued by DMSCLOSE for a file mode number 3 file failed.
WARNING	90617	The minidisk file was erased because it was opened with an intent of REPLACE, but no records were written to it.
ERROR	10000	System error. Attempt to write a block but the file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Severity	Reason Code	Description
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	Storage group space limit exceeded.
ERROR	51100	System error. No minidisks assigned to the storage group.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90129	There are already 2^{31} - 1 blocks in the file, therefore a new logical block is not available.
ERROR	90130	There is not enough free virtual storage available for file system control blocks.
ERROR	90300	Illegal parameter specified. The COMMIT or NOCOMMIT parameter was missing or specified incorrectly, or SHORTDATE, FULLDATE, or ISODATE was specified incorrectly, or extraneous parameters were found.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero *wuerror* length must be at least 12 bytes.
ERROR	90472	Invalid *requestid*. It must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Nonnumeric value in date specification.
ERROR	90498	Invalid time format; must be in the form *hh:mm:ss*.
ERROR	90499	Nonnumeric value in time specification.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.
ERROR	95750	No file opened by DMSOPEN found for the specified token.

DMSCOMM - Commit



Context

Coordinated Resource Recovery: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSCOMM is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCOMM
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCOMM" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCOMM routine to commit (keep) the changes made to the protected resources on a work unit.

DMSCOMM notifies CMS and the resource managers that your application has reached an internal point of consistency and that all protected resources should have their updates made permanent and available to other users of the resources. All changes to protected resources (SFS and non-SFS) are committed when this routine is called. The Coordinated Resource Recovery (CRR) facility provided by CMS makes it possible to commit multiple protected resources. For more information on CRR, see the *z/VM: CMS Application Development Guide*.

Changes made to BFS files are not protected. That is, they do not participate in CRR and are committed independently of other resources, including SFS, on the work unit.

Note: Changes to minidisk files cannot be explicitly committed or rolled back. Changes are committed when the last file on the minidisk that is open for output is closed.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCOMM.

reascode
(output, INT, 4) is a variable for the reason code from DMSCOMM.

workunitid
(input, INT, 4) is a variable for specifying the work unit associated with the DMSCOMM routine. If you want to specify an optional parameter following the *workunitid* parameter without specifying what work unit to commit, specify a value of 0 for the *workunitid* parameter. When you do this, the current default work unit is committed.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning extended error information for the file pools involved in the commit. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the same effect as omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

Usage Notes

1. If DMSCOMM is called and there is a return code of 8, the status of all resources involved remains the same as it was before the commit was issued, so the application can either reissue DMSCOMM or issue DMSROLLB to roll back the work unit. This is also true if the COMMIT parameter is specified on a different CSL routine.
2. If your application uses high-level language statements or OS simulation in an assembler program to update data controlled by a resource that participates in CRR, be sure to empty the buffers (for example, close the files) before trying to commit. This may be required because some high-level languages and assembler programs use OS/MVS queued sequential access method (QSAM) for output files. For example, if a program uses OS/MVS QSAM with blocked records for an SFS output file, some of the most recently written output records may not be committed if the program tries to commit without first closing the QSAM file. These records will subsequently be committed when the program closes the file and either commits the work unit or allows end-of-command processing to commit the work unit. Using only CSL routines, FS macros, or the EXECIO command to write to SFS files avoids this situation.
3. If a file or directory is open for a work unit and DMSCOMM is issued, the file remains open and all changes are committed (see Usage Note "2" on page 72 for exceptions). Current read and write pointers are not changed. There are, however, some situations that can prevent a commit:
   - Files that have been modified using the DMSWRBLK (Write Blocks) routine remain open on the work unit.
   - Catalogs (opened by the DMSOPCAT (Open Catalog) routine) remain open.
4. When working with a file with the INPLACE attribute, DMSCOMM makes some file updates available to concurrent readers. See "DMSOPEN - Open" on page 340 for more information.
5. DMSCOMM commits all protected resources. If your application accesses other resources that do not participate in CRR, it must commit those resources using the commit interface of those resources.

   Changes made to BFS files are not protected. That is, they do not participate in CRR and are committed independently of other resources, including SFS, on the work unit.
6. The setting of the synchronization point options dictates whether CMS will wait for CRR resynchronization to complete before returning to the application. See "DMSSSPTO - Set Synchronization Point Options" on page 508 for more information.
7. Certain errors can cause the work unit to be rolled back. See the description in the "Reason Codes" table that follows for these errors.
8. You can use "DMSGETSP - Get Synchronization Point Errors" on page 277 to get more detailed information about errors or warnings that occurred during the commit. If you are using a protected conversation, error data may be created even for return code 0. To get this error information, use "DMSPCAER - Protected Conversation Adapter Errors" on page 355.
9. DMSCOMM's error reason codes can also be returned by other SFS-related CSL routines when they complete their operation successfully but cannot commit the work unit. In this case, the reason code is 50500, and if the *wuerror* parameter was used when the routine was called, one of DMSCOMM's reason codes is returned in the *error_reascode_info* field of one of the FPERROR entries (see "DMSWUERR - Work Unit Error Data Deblocker" on page 553) for more information.

In addition to the reason codes for DMSCOMM, the *error_reascode_info* field can contain 50500, to indicate that the user has exceeded his file space.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCOMM, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCOMM.

Note: These reason codes can also be returned by other CSL routines. See usage note “9” on page 72.

WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. SEVERE ERROR means that the state of different resources may be inconsistent. Warnings cause return code 4, errors cause return code 8 or 12, and severe errors cause return code 16 or 20.

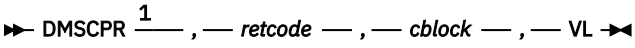
DMSCOMM can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	51060	The user reached or exceeded the SFS file space threshold for one or more file spaces.
WARNING	76070	Changes to non-recoverable files have been lost. This code appears only as part of the work unit error information that is provided when you specify the *wuerror* parameter, or when you request SFS error data using the DMSGETSP (Get Synchronization Point Errors) or DMSGETER (Get My Errors) CSL routines.
WARNING	81054	Resynchronization is in progress on one or more protected resources and their consistency state is unknown. In other words, if the resynchronization is successful, all protected resources will be committed and if it is not successful, changes to all protected resources will be rolled back.
WARNING	81055	All protected resources have been committed. However, the availability of a resource has changed. For example, an open file may have been closed.
ERROR	35000	The work unit was rolled back. System limits were exceeded when attempting to commit. CMS cannot commit more than approximately 230 resources on a work unit.
ERROR	54500	The CRR recovery server log limits were exceeded. The work unit was rolled back.
ERROR	79058	A synchronization point is already in progress for the work unit. This request was ignored.

Severity	Reason Code	Description
ERROR	79061	A resource is not in a state that can be committed. This request was ignored, the work unit was not rolled back or committed, and all protected resources were left in the same state they were prior to the call to DMSCOMM. Reasons this could occur in a protected conversation include: • A protected conversation is not in send, defer, or synchronization point state on a commit function • A protected conversation is in send state but did not finish sending a logical record (applies to basic conversation only) on an attempt to commit. Reasons this could occur with SFS include: • A catalog is open (see DMSOPCAT). • Committing the work would exceed the user's file space limit.
ERROR	81053	The work unit was rolled back; all protected resources have been restored to their prior mutually consistent state. A resource manager (such as SFS) or a conversation partner detected a problem and initiated a rollback.
ERROR	81054	The COMMIT operation failed. A resynchronization operation is in progress to finish rolling back one or more protected resources. Their consistency state is unknown. If the resynchronization is successful, all resources will be rolled back.
ERROR	81056	The CRR recovery server is not available. The work unit was rolled back; all protected resources have been restored to their prior mutually consistent state.
ERROR	81059	The work unit was rolled back due to an application error: Two or more protected conversation partners initiated a commit at the same time.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90540	Invalid work unit ID. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSCOMM.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement. This is an SFS-specific reason code.
ERROR	96100	Insufficient virtual storage. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSCOMM. This request was ignored.

Severity	Reason Code	Description
SEVERE ERROR	81051	A protocol violation may have provoked an inconsistent state among the resources. **RC=16** The COMMIT operation completed. One or more protected resources may have rolled back. **RC=20** The COMMIT operation failed. The work unit was rolled back, but one or more protected resources may have committed.
SEVERE ERROR	81052	Resource manager or operator intervention resulted in an inconsistent state among the resources. **RC=16** The COMMIT operation completed. One or more protected resources have rolled back. **RC=20** The COMMIT operation failed. When trying to roll back changes, one or more protected resources committed.

DMSCPR - Data Compression Services



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Call Format

The format for calling a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSCPR, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCPR
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCPR" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCPR routine to interface with system Data Compression Services.

DMSCPR is a general user interface to system Data Compression Services. It has two defined functions:

- Compressing data
- Expanding previously compressed data

This interface uses the S/390® hardware compression instruction CPMSC. If your hardware does not support the CMPSC compression instruction, the system software simulation of the instruction will be used to perform the service.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCPR.

cblock
(input, INT, 4) is the address of a parameter list for compression services mapped by the CSRYCMPS macro CMPSC DSECT fields. For more detailed information on the CSRYCMPS macro, see *z/VM: CMS Macros and Functions Reference*.

Usage Notes

1. Callers in primary space addressing mode must supply zero values for ALETs in the CBLOCK.
2. The compression and expansion dictionaries specified in the CBLOCK must both be defined on page boundaries and must be contiguous in storage.
3. If calling DMSCPR in AR mode, Access Registers 0, 1, 13, 14, and 15 must be zero. When calling Data Compression Services through the CSL interface, the CBLOCK parameter list cannot be in an address space.
4. When using the direct call format CSL call, the user must issue "GLOBAL TXTLIB VMLIB" before calling DMSCPR.
5. The IBM-supplied CSL template file for the DMSCPR routine is DMSCPRTP TEMPLATE, as shown in Figure 1 on page 77. This template file maps the compression service parameter list that is needed for the CSL interface to the CMPSC hardware compression instruction.

```
DIRECT 2 2          2 parms maximum, 2 required
SBIN     4  OUTPUT  Return Code
SBIN     4  INPUT   CMPSC Block Address
```

Figure 1. DMSCPRTP TEMPLATE File

DMSCPRTP TEMPLATE contains templates for two required parameters:

a. The ***return code*** from DMSCPR:

Value
Meaning

0
The operation was successful.

Any other
The operation was unsuccessful.

b. The CBLOCK parameter contains the address of a 36-byte compression block parameter list to be used by Data Compression Services.

Return Codes

The following table lists the return codes returned by DMSCPR.

Code
Meaning

0
No errors detected.

4
Target operand exhausted before source.

16
An operand is missing.

20
Value in CMPSC_SYMSIZE is not supported. Must be 1-5.

24
No work to do. The compression area length (the target for compression, the source for expansion) is not large enough to hold even one compression symbol.

28
Compression dictionary processing exceeded the limit of 260 for the length of a compressed symbol.

32
A dictionary entry exceeded the limit of 260 total children.

36
A dictionary entry exceeded the limit of a child count of 6.

40
A dictionary entry exceeded the limit of 4 extension characters when there were 0 or 1 children.

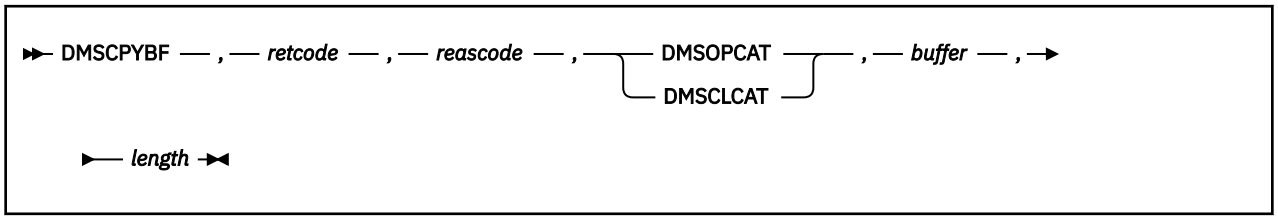
44
A sibling descriptor dictionary entry has a count of 0.

48
Extension of a symbol used more than 127 dictionary entries.

100
Data Compression services are not available.

104
A storage management error occurred either on the obtain or the release of a 144 byte register save area. This storage area is used to save the caller's General Purpose and Access registers over the call to the system Data Compression services routine.

DMSCPYBF - Copy Buffer



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSCPYBF is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCPYBF
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCPYBF routine to get all the information returned on a previous call to the DMSOPCAT (Open Catalog) or DMSCLCAT (Close Catalog) routine by providing a buffer large enough to hold the data. You can use this routine when the buffer specified on DMSOPCAT or DMSCLCAT was too small.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCPYBF.

reascode
(output, INT, 4) is a variable for the reason code from DMSCPYBF.

DMSOPCAT
(input, CHAR, 8) specifies that the previous call was to Open Catalog.

DMSCLCAT
(input, CHAR, 8) specifies that the previous call was to Close Catalog.

buffer
(output, CHAR, *length*) is an area large enough to contain all of the information returned by the preceding Open Catalog (DMSOPCAT) or Close Catalog (DMSCLCAT).

length
(input, INT, 4) is a variable for specifying the length of the preceding character variable (*buffer*). Use the value in the first field of the buffer returned by Open Catalog or Close Catalog.

Usage Notes

1. The call to Copy Buffer must immediately follow a previous Open Catalog or Close Catalog call. If that request was asynchronous, then this call must immediately follow the Check (DMSCHECK) call.
2. The amount of storage needed was returned on the previous call to Open Catalog or Close Catalog. Declare a storage area for that amount and then call this function with the correct buffer length. See

the *buffer* parameter description for Open Catalog and Close Catalog for more information about what information is returned.

Return Codes and Reason Codes

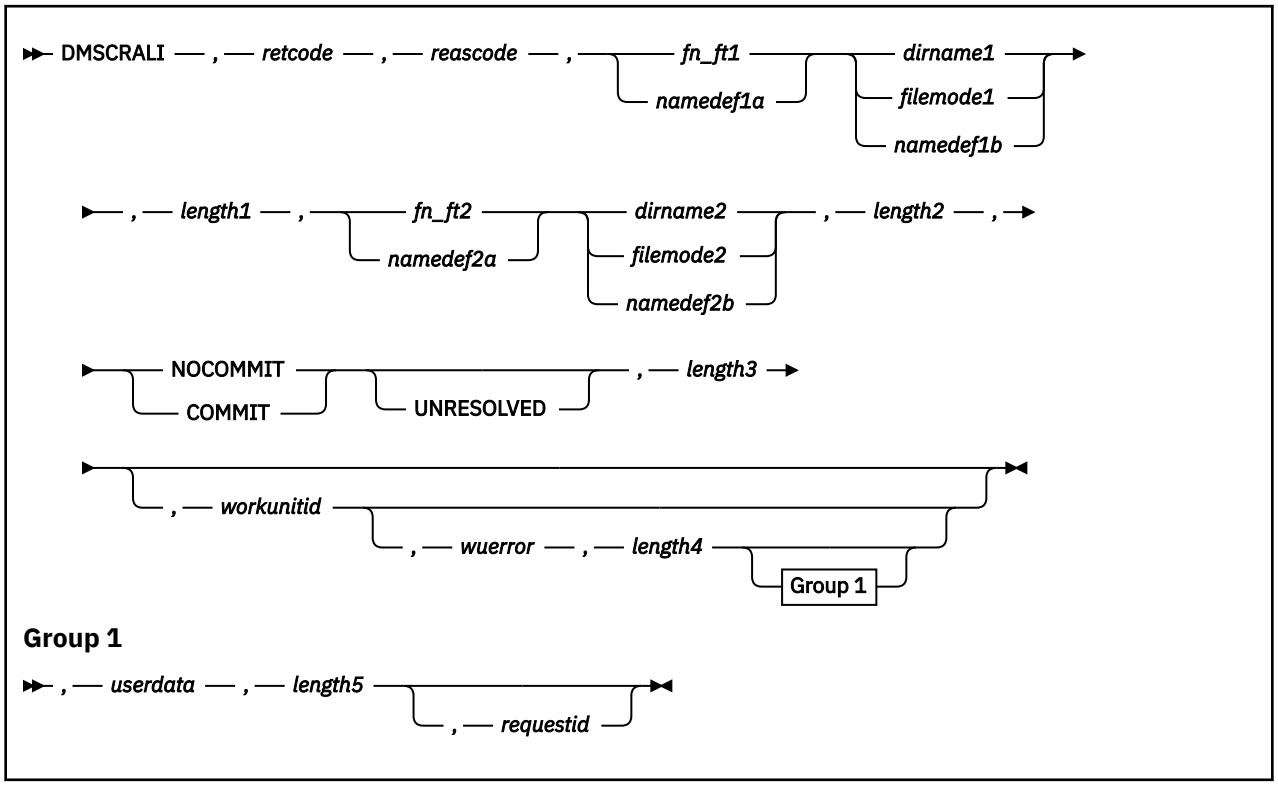
For lists of the possible return codes from DMSCPYBF, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCPYBF. ERROR indicates that the request failed. Errors cause return code 8 or 12.

DMSCPYBF can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90276	No response data available for the previous Open Catalog or Close Catalog.
ERROR	90479	Invalid keyword specified, must be DMSOPCAT or DMSCLCAT.
ERROR	90485	Invalid buffer length specified.

DMSCRALI - Create Alias



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSCRALI is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCRALI
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCRALI routine to create an additional name (alias) for a file in an SFS directory. You can create the alias in any file control directory for which you have write authority (see usage note "4" on page 83).

The new file is known as an alias, while the original file is known as the base file. Once an alias has been created, the alias name becomes a pointer to the base file that contains the data. The alias does not contain data of its own. You can also create an alias on an alias.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCRALI.

reascode
(output, INT, 4) is a variable for the reason code from DMSCRALI.

fn_ft1
(input, CHAR, 3-17) is a variable for specifying the file name and file type for which an alias is being created. The issuer must be the owner of the file or must have been granted read or write authority to it.

namedef1a
(input, CHAR, 1-16) is a variable for specifying the namedef of the file name and file type of the file for which an alias is being created. The issuer must be the owner of the file or must have been granted read or write authority to it.

dirname1
(input, CHAR, 1-153) is a variable for specifying the name of the directory that contains the file for which the alias is being created.

filemode1
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory that contains the file for which the alias is being created.

If a one character namedef is used in this field, it is treated as a namedef rather than as a file mode letter. If the file mode letter is not for an accessed SFS directory, an error return code results.

namedef1b
(input, CHAR, 1-16) is a variable for specifying a namedef of the directory or a namedef of the file mode of the file for which the alias is being created.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft1* or *namedef1a* together with *dirname1*, or *filemode1*, or *namedef1b*, and separating blanks). See "Compound Variables" on page 15 for coding details.

fn_ft2
(input, CHAR, 3-17) is a variable for specifying the name of the alias.

namedef2a
(input, CHAR, 1-16) is a variable for specifying the *namedef* of the file name and file type of the alias.

dirname2
(input, CHAR, 1-153) is a variable for specifying the name of the directory into which the alias will be put. The issuer must own or have write authority on the directory specified by *dirname2* or *namedef2b*.

filemode2
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory that alias will be put into.

If a one character namedef is used in this field, it is treated as a namedef rather than as a file mode letter. If the file mode letter is not for an accessed SFS directory, an error return code results.

namedef2b
(input, CHAR, 1-16) is a variable for specifying the *namedef* of the directory into which the alias will be put.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft2* or *namedef2a* together with *dirname2*, or *filemode2*, or *namedef2b*, and separating blanks). See "Compound Variables" on page 15 for coding details.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

UNRESOLVED
(input, CHAR, 10) causes DMSCRALI to create an unresolved alias if the base file does not exist or the correct authorizations to the directory and base file do not exist. A warning reason code is issued. See usage note "4" on page 83.

Administrator authority is required to use UNRESOLVED.

length3
(input, INT, 4) is a variable containing the length of the preceding character parameter (COMMIT or NOCOMMIT, and UNRESOLVED).

workunitid
(input, INT, 4) is the work unit ID to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended error information for the file pools involved in the commit. See *wuerror* under "Common Parameters" on page 15 for more information.

If it is specified, it must be followed by a length field.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see usage notes "13" on page 84 to "15" on page 84).

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length5* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Usage Notes

1. If a user has a file in directory1, and he also wants to reference that file through directory2, he can create an alias in directory2 to the file in directory1. The user can also create an alias to a file in the same directory, thus creating a synonym for the file in the same directory.
2. If user1 needs to share user2's file, user2 can grant user1 read or write authority on the file. User1 can then create an alias to that file in any of his directories. An alternate way to accomplish the same thing is for user1 to grant user2 write authority on a particular directory. User2 may then create an alias for the file to be shared in user1's directory.
3. If user1 puts an alias into user2's directory for a base file owned by user3, both user1 and user2 must have authority to the base file.
4. To create an alias:
   - The target directory must be file control.
   - You must have authority to write to the target directory or specify UNRESOLVED.

- You must have explicit authority to the base file (the implicit authority of an administrator is not sufficient) or specify UNRESOLVED.
- The owner of the target directory must own or have explicit authority to the base file.

Explicit authority means authority to read or write to the file granted with the CMS GRANT AUTHORITY command or the DMSGRANT CSL routine.

5. Unresolved aliases are intended to improve the performance of applications that restore file pools and storage groups from backup data. See the *z/VM: CMS File Pool Planning, Administration, and Operation*.
6. UNRESOLVED has no effect when the base file exists and the correct authorizations exist: an alias is created (rather than an unresolved alias).
7. FILELIST and LISTFILE display unresolved aliases like revoked aliases, but they are different: a revoked alias can never be resolved, while an unresolved alias can be resolved into an alias when the base file is restored.
8. If a base file is erased, any alias for that file is changed to an erased alias.

   If the explicit authority you have to a base file is revoked, any alias you have for that file is changed to a revoked alias. Authority to a base file can be READ or WRITE to a file in a file control directory, or DIRREAD or DIRWRITE to a directory control directory.
9. Aliases cannot cross file system file pools.
10. The alias has the same file mode number as the base file.
11. When you refer to an alias, you are actually referring to the base file.
12. The following tables depict the situations for creating an alias on a locked file or directory using DMSCRALI. *Yes* means the user can create an alias on a file or directory with the given lock mode. *No* means the user cannot.

User	Lock Mode on the Base File		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Lock Creator	Yes	Yes	Yes
Another User	Yes	Yes	No

User	Lock Mode on the Target Directory		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Lock Creator	No	Yes	Yes
Another User	No	No	No

13. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file for which you are creating an alias.
14. If you call DMSCRALI with the *userdata* parameter for an ESM-protected file, but you do not have the proper authority for the file or specify an invalid value in *userdata*, the routine fails and returns an error reason code.
15. If you call DMSCRALI without the *userdata* parameter (because the ESM does not require it) for an ESM-protected file, the routine succeeds, but a warning reason code is returned.
16. Aliases cannot reside in a directory control directory. Use DMSEXIDI (SFS Exist - Directory) or DMSGETDD (SFS Get Directory - Dir) to retrieve the directory attribute.
17. If the COMMIT parameter is specified and the return code is 8, then either:

    - An error occurred during the processing of the Create Alias operation, or

- The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the commitment failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

18. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
19. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCRALI, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSCRALI. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

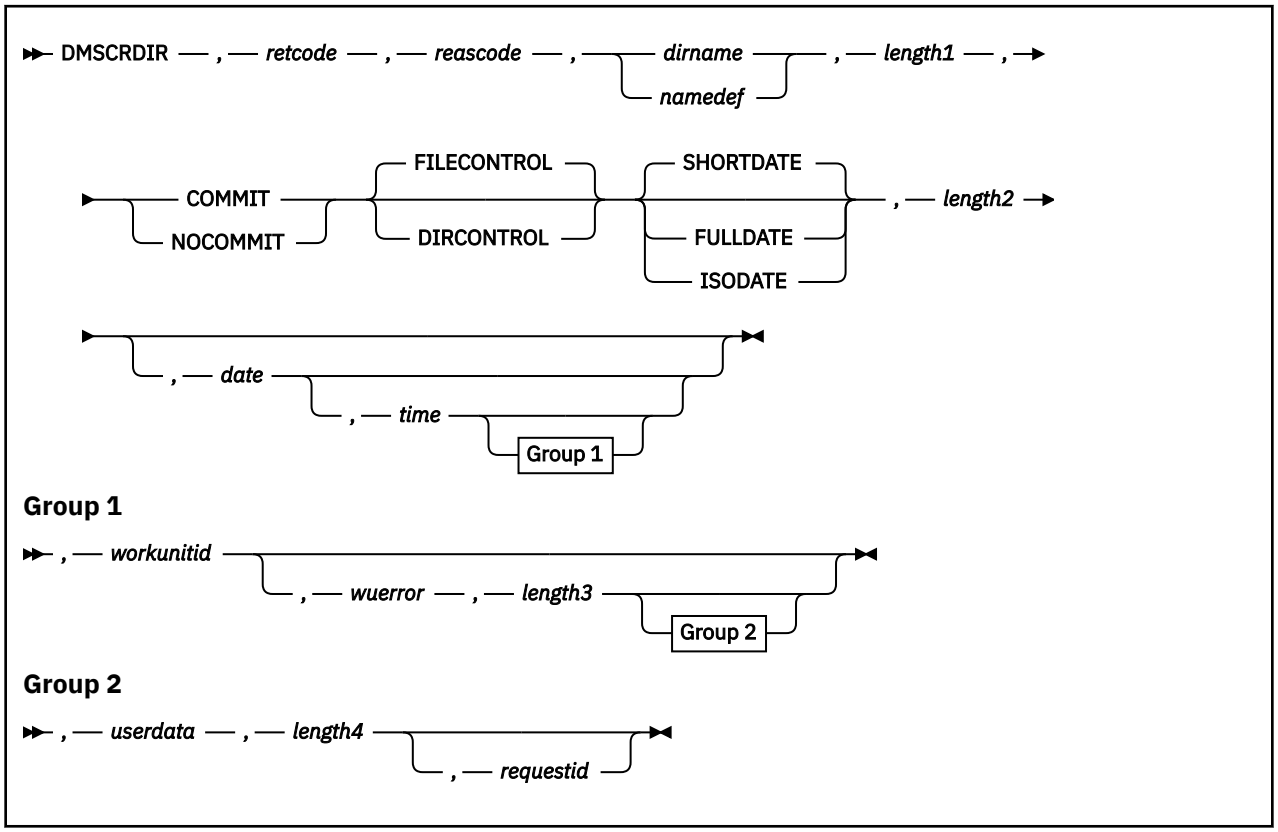
DMSCRALI can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	The file space warning threshold was reached or exceeded for one or more file spaces while committing or rolling back a work unit.
WARNING	61620	An unresolved alias was created.
WARNING	98700	The UNRESOLVED keyword is not supported by this file pool. However, if the base file and the necessary authorizations exist, the alias was created.
ERROR	10000	System error. COMMIT was specified. SFS attempted to write a block, but the file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Alias name already exists as a base file or an alias in the same directory.
ERROR	44000	Attempt to create an alias failed because: • The base file, its parent directory, or the target directory does not exist. • You do not have explicit read or write authority to the base file. • You do not have write authority to the target directory. • The owner of the target directory does not have read or write authority to the base file.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the commitment failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Severity	Reason Code	Description
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	63800	Specified directory is a directory control. Aliases cannot be created in directory control directories.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back (COMMIT option only).
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90250	The file name and file type (or *namedef*) are required but were not specified.
ERROR	90300	Invalid parameter in CSL parameter list. Must be COMMIT or NOCOMMIT.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive options.
ERROR	90330	Duplicate options in CSL parameter list. COMMIT or NOCOMMIT or UNRESOLVED was specified more than once.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.
ERROR	90380	Missing parameter in CSL parameter list. COMMIT or NOCOMMIT must be specified.
ERROR	90410	Invalid length specified for *userdata*, file ID, or keyword (COMMIT, NOCOMMIT, UNRESOLVED) parameters.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type is longer than 8 characters or contains an invalid character.
ERROR	90450	Global characters (* or %) were found in either the file name or file type.
ERROR	90460	File pool IDs in source and target directory names are not the same.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is invalid.
ERROR	90505	The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* parameter is longer than 16 characters.

Severity	Reason Code	Description
ERROR	90530	The namedef part of the *fileid* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a file ID namedef was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	90611	Input file mode letter did not represent an accessed SFS directory in the second file ID.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSCRDIR - Create Directory



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSCRDIR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCRDIR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCRDIR routine to create an SFS directory once you have been enrolled as a user in a file pool.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCRDIR.

reascode
(output, INT, 4) is a variable for the reason code from DMSCRDIR.

dirname
(input, CHAR, 2-153) is a variable for specifying the name of the directory being created.

namedef
(input, CHAR, 1-16) is a variable for specifying a namedef for the directory.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*dirname* or *namedef*).

COMMIT
(input, CHAR, 6) means keep all changes made during a work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

DIRCONTROL
(input, CHAR, 10) sets the directory as a directory-control directory.

FILECONTROL
(input, CHAR, 11) sets the directory as a file-control directory. FILECONTROL is the default.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where *yy* is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameters (COMMIT or NOCOMMIT; FILECONTROL or DIRCONTROL, if specified; and SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding detail.

date
(input, CHAR, 8 or 10) is the date of last update for the directory. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. If the date is omitted, the current date is used.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you omit the *date* variable, or leave blanks, the system generates the date using local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the time of last update for the directory, in the format: *hh:mm:ss*. If the time is blanks or is omitted, the current time is used.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. Locking a directory UPDATE or EXCLUSIVE prevents anyone except the holder of the lock from creating a subdirectory in it. No one can create a subdirectory in a directory that is locked SHARE.
2. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the directory that you are creating.
3. Some characteristics of file control directories are:

   - All requests for file control directories and the files within them are directed to the file pool server machine. CMS uses Advanced Program-to-Program Communication/VM (APPC/VM) to communicate with the server. The server checks authorization and implicitly locks necessary resources whenever a file is opened.
   - Any number of users can access a file control directory in read/write mode. These users can write to different files within the directory at the same time. SFS ensures that no two users write to the same file at the same time.

     Any number of users can access file control directories in read-only mode. Authorized users can read files that are concurrently being written to by some other user. SFS ensures that the readers see a consistent version of the file from the time it is opened until the time it is closed. Except for files having the INPLACE attribute, readers see committed changes when they close and reopen a file. Readers can see changes to INPLACE files as they are written.
   - Users cannot access the directory unless they have either READ or WRITE authority for that directory.
   - The directory can contain aliases that refer to files in file control or directory control directories.
   - A file in a file control directory can be the source or target of a DMSRELOC (Relocate) routine.

- Directories can be relocated from or to a directory with the file control attribute, even if the directory is accessed.
- The directory and its files can be locked UPDATE, SHARE, or EXCLUSIVE.
- READ and WRITE authority can be granted on file control directories or on individual files within file control directories. NEWREAD and NEWWRITE authority can also be granted on file control directories. See the DMSGRANT (Grant Authority) routine for more information about these authorities.
- The creator (owner) of a file control directory has WRITE and NEWWRITE authority by default.
- If you have a directory with the file control attribute accessed read-only, you can create a subdirectory therein.

4. Directory control directories have operational and performance characteristics that differ significantly from file control directories. Characteristics of directory control directories include:

   - Some performance benefits are possible with directory control directories.

     When the directory is accessed read-only by a local user, using the CMS ACCESS command, it is possible that its contents and files may be assigned to an available *data space*. A data space is an area of virtual storage that can be addressed by your virtual machine. When a data space is used, your virtual machine avoids the processing overhead of communicating with the file pool server.

     An exception is that all requests to read INPLACE files go through the server, even if the directory is in a data space.

     In addition, authorization checking and locking occur when the directory is accessed rather than each time a file is opened, as is the case with file control directories.
   - Directory control directories do not allow the same level of concurrency that file control directories do. While read concurrency is the same, write activity is restricted. In particular, when a user has a directory control directory accessed in read/write mode, no other user can write to the directory. If no one has the directory accessed in read/write mode, several users can concurrently write to different files within the directory by using CSL routines to open the files.

     Only one read/write access, using the CMS ACCESS command, is permitted to the directory at a time. Many read-only accesses are permitted concurrently. Refer to the CMS ACCESS command for more information.
   - Those who access the directory in read-only mode, using the CMS ACCESS command, will see file changes only when they release and reaccess the directory. One exception concerns INPLACE files. Readers can see changes to INPLACE files as soon as they are written without having to close and reopen the file or reaccess the directory.
   - Those who access the directory in read/write mode, using the CMS ACCESS command, will see any changes to files in the directory as they are made and committed.

     Note: When you access a directory (read-only) more than once without first releasing it, you will not see any committed changes to the directory until you release all nested accesses. After releasing all accesses, the changes will be available to you when you next access the directory.
   - Users must have either directory control read (DIRREAD) or directory control write (DIRWRITE) authority to access a directory control directory.
   - The directory cannot contain aliases.
   - Aliases in file control directories can be used to read from or write to base files in directory control directories. If you have the directory control directory accessed in read-only mode, however, attempts to write to the base file through the alias will fail.
   - A file in a directory with the directory control attribute cannot be the source or target of a DMSRELOC (Relocate) routine.
   - Directories can be relocated from or become subdirectories of a directory with the directory control attribute. This requires that you do not have the containing directory accessed read-only and that no other user has it accessed read/write.

- You cannot relocate a directory with the directory control attribute if anyone other than you has the directory accessed.
- The directory and its files can only be locked UPDATE.
- No authority other than directory control read (DIRREAD) and directory control write (DIRWRITE) authority can be granted or revoked for the directory or its files.

5. The creator of a directory control directory has directory control write (DIRWRITE) authority by default.
6. To let others use a directory control directory, you must grant either directory control read (DIRREAD) or directory control write (DIRWRITE) authority. For more about these authorities, refer to the DMSGRANT routine.
7. You cannot create a subdirectory in a directory control directory if you have the directory control directory accessed read-only or if some other user has it accessed read/write.
8. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Create Directory operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
9. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
10. If you have an active asynchronous request for a file pool, you cannot issue any other requests except a rollback required for the affected file pool on the specified work unit.
11. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
12. When *date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
13. When *date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the directory.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
14. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCRDIR, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSCRDIR. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

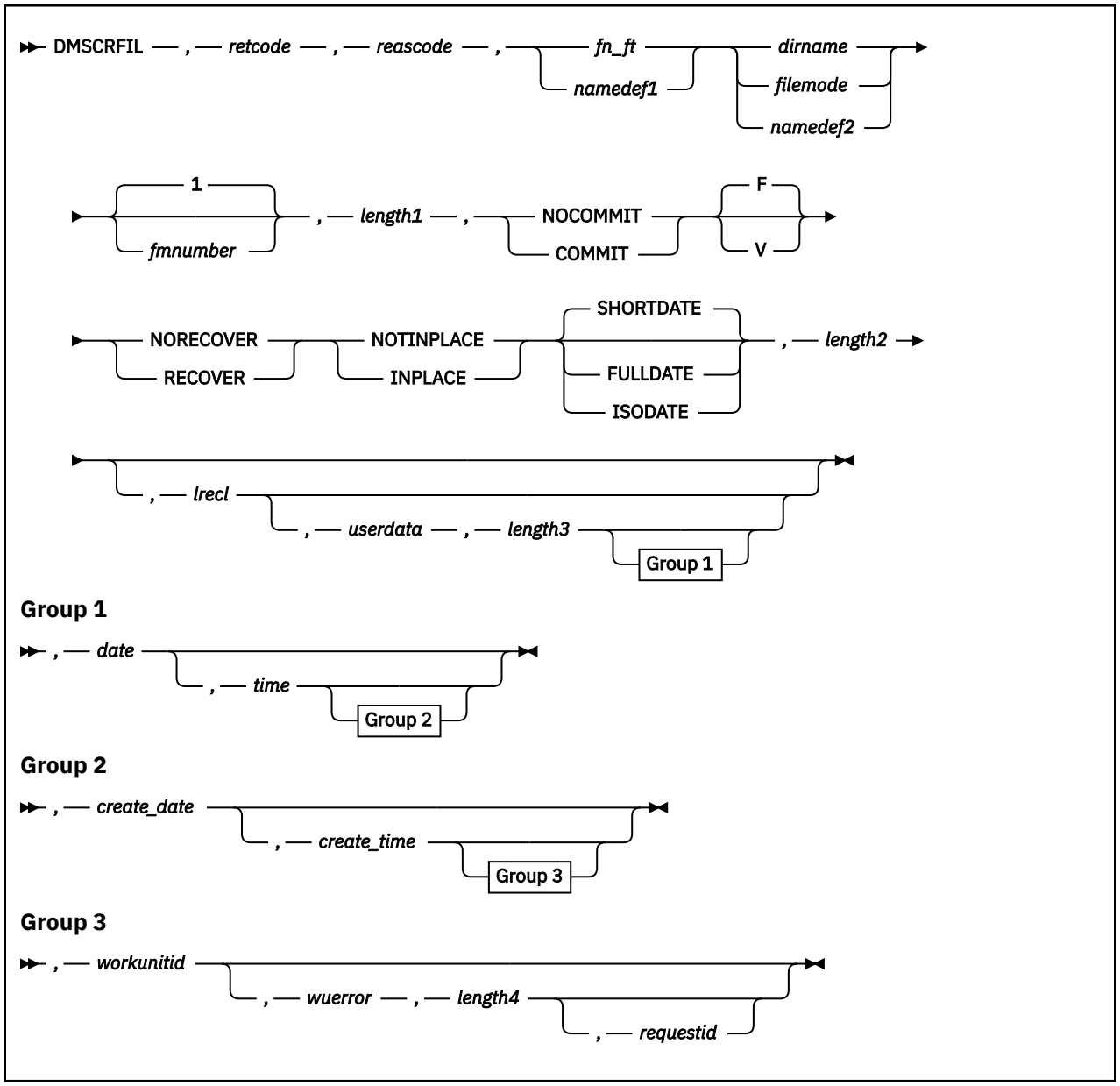
Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

DMSCRDIR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System Error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Directory with the same name already exists.
ERROR	44000	Parent directory does not exist or you are not the owner of it.
ERROR	44100	You are not authorized to create the directory named on this request. Note, this error only occurs when SFS is running with an external security manager.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the commitment failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	60200	Attempt to create top-level directory.
ERROR	63700	Specified parent directory control directory is accessed read-only.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back (COMMIT parameter only).
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90300	Invalid input parameter in CSL parameter list. Valid parameters are COMMIT, NOCOMMIT, FILECONTROL, DIRCONTROL, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, FILECONTROL and DIRCONTROL, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, FILECONTROL, DIRCONTROL, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the dirname parameter.

Severity	Reason Code	Description
ERROR	90380	Missing parameter in CSL parameter list. COMMIT or NOCOMMIT is required.
ERROR	90410	Invalid length specified for the *userdata*, directory ID, or COMMIT/NOCOMMIT parameters.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Nonnumeric value in date specification.
ERROR	90498	Invalid time format; must be in the form *hh:mm:ss*.
ERROR	90499	Nonnumeric value in time specification.
ERROR	90500	The specified dirname is invalid.
ERROR	90505	The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef parameter is longer than 16 characters.
ERROR	90530	The specified namedef does not exist or was used incorrectly. For example, a namedef that was created for a file name/file type was used where a dirname namedef was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and file pool ID was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSCRFIL - Create File



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSCRFIL is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCRFIL
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCRFIL routine to create a new empty file in an SFS directory.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCRFIL.

reascode
(output, INT, 4) is a variable for the reason code from DMSCRFIL.

fn_ft
(input, CHAR, 3-17) contains the file name and file type of the file being created.

namedef1
(input, CHAR, 1-16) is a variable containing a namedef (a temporary name) for the file being created.

dirname
(input, CHAR, 1-153) is a variable specifying the directory containing the file being created.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If a one character namedef is used in this field, it is treated as a namedef rather than as a file mode letter. If the file mode letter is not for an accessed SFS directory, an error return code results.

namedef2
(input, CHAR, 1-16) is a variable containing a namedef (a temporary name) for the directory containing the file being created.

fmnumber
(input, INT, 1) is a variable containing the file mode number to be assigned to the file. The default is 1.

length1
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*fn* or *namedef1*, *dirname* or *namedef2*, and *fmnumber*).

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit. That is, all changes made during a work unit, from either the start of the work unit or from the last request to commit, are kept.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

F
(input, CHAR, 1) indicates that all records in the file must be the same length. This is the default. The record length assigned to the file may be specified in *lrecl*. If *lrecl* is omitted for a fixed length file, the record length will default to 80 bytes.

V
(input, CHAR, 1) indicates that the file will contain variable length records. The record length for a variable length file is initialized to 1 byte when the file is created. It is changed to reflect the length of the longest record in the file.

RECOVER
(input, CHAR, 7) indicates that application initiated rollbacks will cause modifications to this file to be rolled back.

NORECOVER
(input, CHAR, 9) indicates that neither the creation of this file, nor subsequent modifications to it, will be rolled back in the event of an application initiated rollback.

INPLACE
(input, CHAR, 7) indicates that updates to existing file blocks will be done in place. This can reduce DASD utilization and enables readers to see file updates by a concurrent writer. However, this option should be specified with caution because data integrity is not guaranteed for INPLACE files.

NOTINPLACE
(input, CHAR, 10) indicates that file writes are to be shadowed such that readers see a consistent view of the file from Open to Close.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* and *create_date* parameters is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* and *create_date* parameters is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* and *create_date* parameters is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (COMMIT or NOCOMMIT; F or V, if specified; RECOVER or NORECOVER; INPLACE or NOTINPLACE; and SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

lrecl
(input, INT, 4) defines the logical record length of the file. The meaning and defaults of this parameter are dependent upon the Record Format that is specified as F or V above.

userdata
(input, CHAR, 1-80) is a variable containing a string of up to 80 characters of user data to be passed to an external security manager (ESM). The format and meaning of the data is defined by the ESM (see the Usage Notes).

length3
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*userdata*). The value of *length3* must not be greater than 80.

date
(input, CHAR, 8 or 10) is the date-of-last-update of the file. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you omit the *date* variable, or leave blanks, the system generates the date using local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the time-of-last-update for the file. This is a character variable with a length of 8 in the form *hh:mm:ss*.

You can have the system determine the time by omitting the parameter or by specifying 8 blanks. The system uses local time.

Use a colon (:) as the separator character and 2 digits in each position. For example:

```
15:09:17.
```

create_date
(input, CHAR, 8 or 10) is the Coordinated Universal Time (UTC) date of creation. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the creation date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *create_date* variable, or leave blanks, the system generates the creation date.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

create_time
(input, CHAR, 8) is the Coordinated Universal Time (UTC) of creation. This is a character variable with a length of 8 in the form *hh:mm:ss*.

You can have the system determine the creation time by omitting the parameter or by specifying 8 blanks.

Use a colon (:) as the separator character and 2 digits in each position. For example:

```
15:09:17.
```

workunitid
(input, INT, 4) is a variable containing the work unit to be used to go to a file pool. This must be a signed integer variable with a length of 4. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) contains the information necessary to have CMS return extended error information. If it is specified, it must be followed by a length field.

length4
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable containing a 0 (for synchronous) or a 1 (for asynchronous) processing of the request. The default is 0. On return, the request ID is an identifier that is used as input on the DMSCHECK CSL routine to complete the Create File request.

Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Usage Notes

1. The resultant file will be empty. It will have zero records and zero file blocks allocated to it.
2. You must have write authority on the directory to issue DMSCRFIL.
3. Users with NEWREAD or NEWWRITE authority to a directory are granted appropriate READ or WRITE authority to new files.
4. The logical record length (*lrecl*) for variable length record files is dynamically updated by CMS file system processing.
5. Default values for the recoverability and overwrite attributes are determined by whether both of them, just one, or neither of them is specified. The initial default is RECOVER and NOTINPLACE, but this default can be changed by the PUSH ATTRIBUTES command or the DMSPUSHA - SFS Push Attributes CSL routine.
6. INPLACE and RECOVER attributes are mutually exclusive.
7. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the object you are erasing.
8. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Create File operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
9. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
10. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
11. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to both the *date* and *create_date* parameters.
12. When *date* and *create_date* are specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
13. When *date* and *create_date* are specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
14. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCRFIL, see Appendix D, "Return Codes," on page 597.

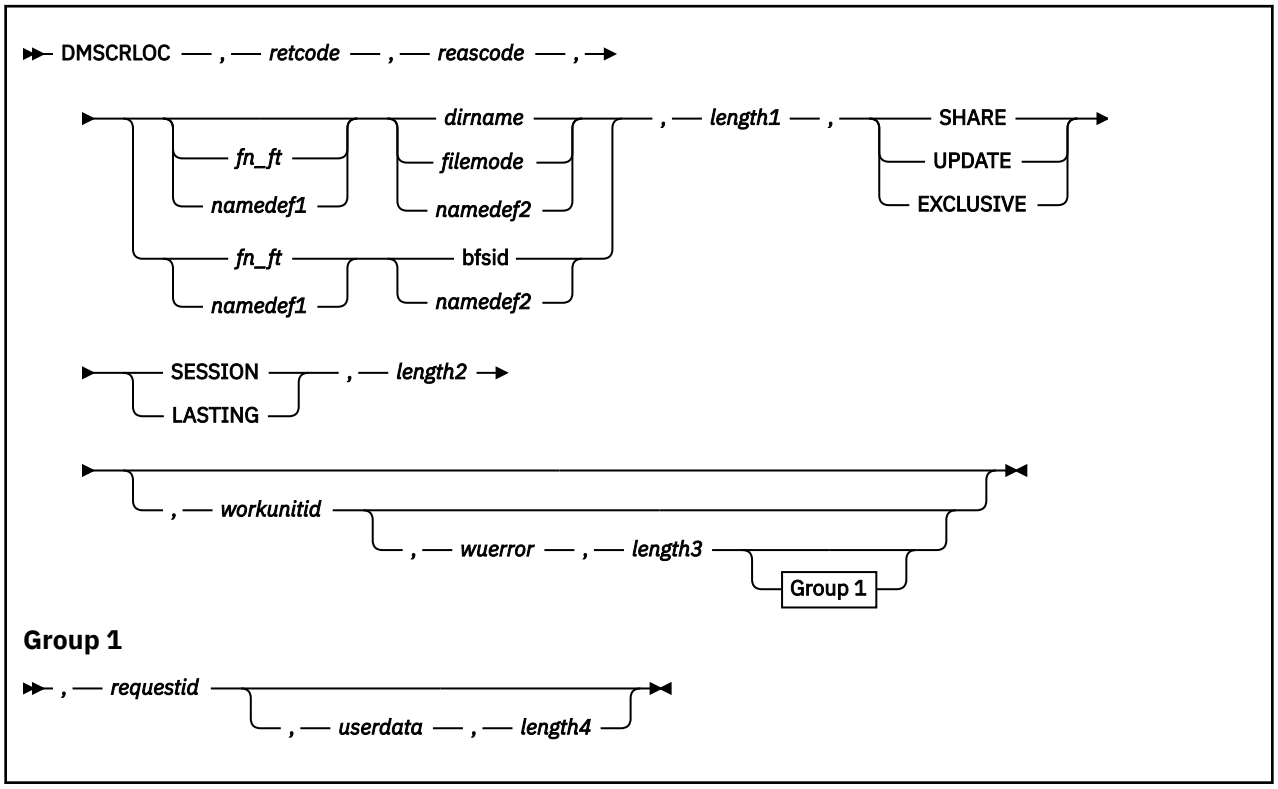
The following table lists the special reason codes returned by DMSCRFIL. ERROR means the request failed, or the request was unsuccessful. Errors cause return code 8 or 12.

DMSCRFIL can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	20000	File already exists.
ERROR	30500	RECOVER and INPLACE attributes are mutually exclusive.
ERROR	31000	Mixing recoverable and nonrecoverable updates for the same file is incorrect within a single work unit.
ERROR	44000	Directory not found, or issuer is not authorized to write to it.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	50600	The user has no storage blocks allocated to his file space.
ERROR	63700	The directory is a directory control directory, accessed R/O by the issuer.
ERROR	76000	Error in commit processing.
ERROR	90250	File name and file type (or *namedef*) are required, but were not specified.
ERROR	90307	Incorrect *lrecl* specified.
ERROR	90310	Incorrect option in CSL parameter list. Valid options are COMMIT or NOCOMMIT, F or V, NORECOVER or RECOVER, INPLACE or NOTINPLACE, and SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, F and V, if specified, are mutually exclusive parameters, RECOVER and NORECOVER are mutually exclusive parameters, INPLACE and NOTINPLACE are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, RECOVER, NORECOVER, INPLACE, NOTINPLACE, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens. There must be more than one token, because *fileid* and *dirname* are required.
ERROR	90380	Required parameter omitted in CSL parameter list.
ERROR	90410	Incorrect length specified for a character variable.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Severity	Reason Code	Description
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than eight characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than eight characters or contains an incorrect character.
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	Incorrect *requestid*.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Incorrect *create_date* or date specified.
ERROR	90498	Incorrect *create_time* or time format.
ERROR	90499	Incorrect *create_time* or time specified.
ERROR	90500	The specified *dirname* is incorrect.
ERROR	90505	The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	Syntactic error in *namedef*.
ERROR	90530	Incorrect *namedef* specified.
ERROR	90540	Specified *workunitid* is incorrect.
ERROR	90590	There is no default file pool currently defined, and file pool ID was not specified as part of *dirname*.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file or directory is open and the file pool server does not have the Commit Without Close enhancement. This is an SFS-specific reason code.

DMSCRLOC - Create Lock



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSCRLOC is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCRLOC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCRLOC routine to create an explicit lock on:

- An SFS file or directory
- A BFS regular file

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCRLOC.

reascode
(output, INT, 4) is a variable for the reason code from DMSCRLOC.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file that is to be locked. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the SFS directory name. If *fn_ft* or *namedef1* is specified, this is the directory containing the file that is to be locked. If neither *fn_ft* nor *namedef1* is specified, this is the directory that is to be locked.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed SFS directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode. If the file mode is not an accessed SFS directory, an error return code will result.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system containing the file that is to be locked.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, if specified; plus *dirname*, *filemode*, *bfsid*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

SHARE
(input, CHAR, 5) specifies that others can read while you read. Many users can have the SHARE lock at the same time. This allows implicit share locks for the same file within the file pool. Locking a file or directory SHARE prevents anyone from changing that file or the files in that directory.

BFS files may not be locked SHARE.

UPDATE
(input, CHAR, 6) specifies that others may only read while you read or update. Only one user can have the update lock at a time. This allows implicit share locks for the same file within the file pool. Locking a file or directory UPDATE prevents anyone other than the holder of the lock from changing that file or the files in that directory.

If UPDATE is specified for a BFS file, the effect is the same as specifying EXCLUSIVE.

EXCLUSIVE
(input, CHAR, 9) prevents others from getting any of the locks or using the file or directory. You cannot get an exclusive lock if there is any activity on the file or directory. Locking a file or directory EXCLUSIVE prevents anyone other than the holder of the lock from changing that file or the files in that directory.

SESSION
(input, CHAR, 7) specifies that the lock is to be removed whenever you issue a Delete Lock (DMSDELOC), terminate a CMS session, terminate a file pool server, or disconnect from the file pool containing the locked object.

LASTING
(input, CHAR, 7) specifies that the lock is to be removed only when a DMSDELOC is issued. This lock will last across CMS sessions and file pool server startups.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (SHARE or UPDATE or EXCLUSIVE, and SESSION or LASTING). See "Compound Variables" on page 15 for coding details.

workunitid
(input, INT, 4) is a variable for identifying the work unit associated with the Create Lock routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

Usage Notes

Common Notes for SFS and BFS

1. DMSCRLOC is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it completes, DMSCRLOC will cause a noncoordinated commit to be done for the work in the affected file pool.
2. A lock created with this routine will be overridden by a Disable Storage Group (DMSDISSG) or Disable Filespace (DMSDISFS).
3. Some batch facilities may be set up to run batch jobs under the user ID of the virtual machine (user) submitting the job. Your system administrator can tell you if the batch facility you are using is set up that way. If it is, the batch machine runs your job under your user ID. Therefore, if you have placed an explicit lock on a file or directory, the job running on the batch machine can still use that file or directory. However, if your batch job has a file open for update, any job you are running (from your user ID) will be barred from updating that file until the batch job is finished with it. Note that the batch facility provided with z/VM does not run under the user ID of the virtual machine submitting the job, but rather under its own user ID.
4. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
6. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file or directory you are locking.

Notes for SFS Only

1. For file control directories and files within them, READ authority is required to create a SHARE lock. WRITE authority is required to create an UPDATE or EXCLUSIVE lock.

   For directory control directories and files within them, DIRWRITE authority is required to create an UPDATE lock, which is the only lock allowed.
2. When DMSCRLOC is issued against an alias, the base file of the alias is used to determine whether the request to create a lock will succeed.
3. You can create only UPDATE locks on directory control directories and the files within them.
4. You can lock a file in a directory control directory when:
   - You have the directory accessed, read-only or read/write
   - Another user has another file locked.

   You cannot lock a file in a directory control directory when another user has:
   - The directory accessed in read/write mode
   - A lock on the directory.
5. You can lock a directory control directory if no one, including yourself, has it accessed WRITE, no one else has it or a file in it locked, and you do not have it accessed READ.

   If your application has obtained a work unit ID on behalf of a virtual machine other than the one it is running in, it is not allowed to create a session lock on an SFS resource for that virtual machine. Any attempt is rejected with an error reason code. You *can* create a lasting lock for another user. (See the *userid* parameter of "DMSGETWU - Get Work Unit ID" on page 280.)
6. The following tables show when it is possible for DMSCRLOC to create a lock on a file when its directory is already locked, or to create a lock on a directory when a file within is already locked. *Yes* indicates that the issuer can create the specified lock under the given set of circumstances.

Issuer has lock on file and requests another on directory:

Lock Mode Requested for Directory	**Lock Mode Already on File**		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Explicit Share	Yes	No	No
Explicit Update	Yes	Yes	Yes
Explicit Exclusive	Yes	Yes	Yes

Issuer has lock on directory and requests another on file:

Lock Mode Requested for File	**Lock Mode Already on Directory**		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Explicit Share	Yes	Yes	Yes
Explicit Update	No	Yes	Yes
Explicit Exclusive	No	Yes	Yes

Issuer requests lock on directory while another user has lock on file:

Lock Mode Requested for Directory	**Other User's Lock Mode on File**		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Explicit Share	Yes	No	No
Explicit Update	Yes	No	No

Lock Mode Requested for Directory	Other User's Lock Mode on File		
	Explicit Share	Explicit Update	Explicit Exclusive
Explicit Exclusive	No	No	No

Issuer requests lock on file while another user has lock on directory:

Lock Mode Requested for File	Other User's Lock Mode on Directory		
	Explicit Share	Explicit Update	Explicit Exclusive
Explicit Share	Yes	Yes	No
Explicit Update	No	No	No
Explicit Exclusive	No	No	No

Notes for BFS Only

1. You must have file pool administration authority to lock a BFS file.
2. You cannot use DMSCRLOC to lock a byte file system (BFS top directory). To lock a byte file system, use the DMSDISFS routine to disable the BFS file space.
3. BFS files may not be locked SHARE. Therefore, a BFS file cannot be locked if it is in use.
4. An UPDATE lock on a BFS file has the same effect as an EXCLUSIVE lock.
5. An EXCLUSIVE lock on a BFS file creates a busy file condition. Only the holder of the lock can access the file.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCRLOC, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSCRLOC. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

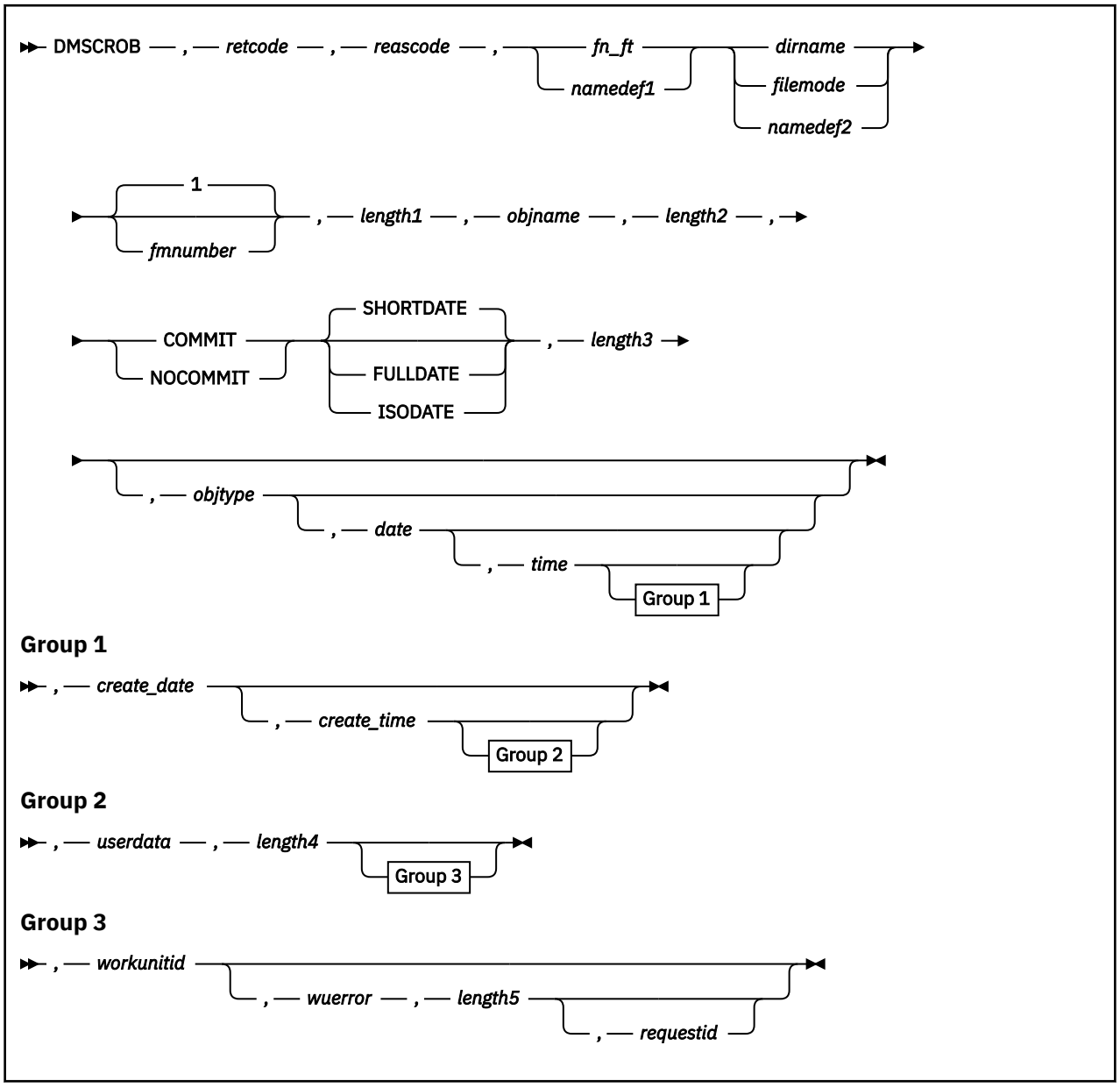
DMSCRLOC can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	02080	You already hold the requested lock.
ERROR	02400	You already hold a SHARE lock on the object and you have requested an UPDATE or EXCLUSIVE lock.
ERROR	02500	You already hold an UPDATE lock on the object and you have requested a SHARE or EXCLUSIVE lock.
ERROR	02600	You already hold an EXCLUSIVE lock on the object and have requested a SHARE or UPDATE lock.
ERROR	02700	Another user holds a SHARE lock on the object and you have requested an UPDATE or EXCLUSIVE lock.
ERROR	02800	Another user holds an UPDATE lock on the object.
ERROR	02900	Another user holds an EXCLUSIVE lock on the object.
ERROR	30000	You do not have the file pool administration authority required to lock a BFS file.

Severity	Reason Code	Description
ERROR	44000	The file pool object to be locked does not exist or you are not authorized to lock it in the requested mode.
ERROR	63900	You cannot lock a file in a directory control directory SHARE or EXCLUSIVE.
ERROR	64000	You cannot lock a directory control directory SHARE or EXCLUSIVE.
ERROR	65400	Object is not a BFS regular file. Only regular BFS files can be locked.
ERROR	66300	SFS has rejected a request for a session lock on behalf of another user.
ERROR	69200	You cannot lock a BFS file SHARE.
ERROR	69300	Object is not a BFS regular file. You cannot create a lock on a BFS directory.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90300	Invalid lock description parameter - (SHARE, UPDATE, or EXCLUSIVE, and SESSION or LASTING).
ERROR	90320	Conflicting parameters in CSL parameter list. The following options are mutually exclusive: UPDATE vs. EXCLUSIVE vs. SHARE SESSION vs. LASTING
ERROR	90330	Duplicate parameters in CSL parameter list. UPDATE, EXCLUSIVE, SHARE, SESSION, LASTING was specified more than once.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.
ERROR	90380	Missing parameter in CSL parameter list. One of each of the following sets of options must be specified: • UPDATE or EXCLUSIVE or SHARE • SESSION or LASTING
ERROR	90410	Invalid length specified for the *fileid*, *dirname*, *userdata*, or lock description (UPDATE, EXCLUSIVE, or SHARE, and SESSION or LASTING) parameter.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.
ERROR	90440	The specified file mode is invalid. It must be an alphabetic character.

Severity	Reason Code	Description
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified director name is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef containing a file name and file type was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95400	The specified work unit was already active for the specified file pool when DMSCRLOC was executed.

DMSCROB - Create External Object



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSCROB is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCROB
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCROB" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCROB routine to create an external object in an SFS directory.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSCROB.

reascode
(output, INT, 4) is a variable for the reason code from DMSCROB.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the external object.

namedef1
(input, CHAR, 1-16) is a variable containing a namedef (a temporary name) for the file name and file type of the external object being created.

dirname
(input, CHAR, 1-153) is a variable specifying the directory containing the external object being created.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If a one character namedef is used in this field, it is treated as a namedef rather than as a file mode letter. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable containing a namedef (a temporary name) for the directory containing the external object.

fmnumber
(input, INT, 1) is a variable containing the file mode number to be assigned to the external object. The default is 1.

length1
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameters (*fn_ft* or *namedef1*, *dirname* or *namedef2*, and *fmnumber*).

objname
(input, CHAR, 1-255) is a variable containing a string characters (referred to as a remote name). The meaning of the remote name is determined by the application the creates the external object. SFS does not check the validity of these characters.

length2
(input, INT, 4) is a variable containing the length of *objname*. It cannot be greater than 255.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit. That is, all changes made during a work unit, from either the start of the work unit or from the last commit, are kept.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* and *create_date* parameters is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* and *create_date* parameters is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* and *create_date* parameters is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length3
(input, INT, 4) is a variable containing the length of the preceding parameter (COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

objtype
(input, CHAR, 8) is a variable containing the object type. The format and meaning of the data in this field are defined by the program querying the external object. The first 3 characters, however, are designed to be used as a product identifier. To ensure uniqueness, this identifier is assigned by IBM. For more information about these identifiers, see the information about the ANCHOR macro in the *z/VM: CMS Macros and Functions Reference*.

If you want to specify an optional parameter following *objtype* without using the *objtype* parameter, put blanks in *objtype* (see usage note "5" on page 113).

date
(input, CHAR, 8 or 10) is the date-of-last-update of the external object. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *date* variable, or leave blanks, the system generates the date using local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the time-of-last-update for the external object. This is a character variable with a length of 8 in the form *hh:mm:ss*.

You can have the system determine the time by omitting the parameter or by specifying 8 blanks. The system uses local time.

Use a colon (:) as the separator character and 2 digits in each position. For example:

```
15:09:17.
```

create_date
(input, CHAR, 8 or 10) is the Coordinated Universal Time (UTC) date of creation. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the creation date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *create_date* variable, or leave blanks, the system generates the creation date.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

create_time
(input, CHAR, 8) is the Coordinated Universal Time (UTC) of creation. This is a character variable with a length of 8 in the form *hh:mm:ss*.

You can have the system determine the creation time by omitting the parameter or by specifying 8 blanks.

Use a colon (:) as the separator character and 2 digits in each position. For example:

```
15:09:17.
```

userdata
(input, CHAR, 1-80) is a variable containing a string of characters to be passed to an external security manager (ESM). The format and meaning of the characters are defined by the ESM.

length4
(input, INT, 4) is a signed integer variable containing the length of the preceding parameter, *userdata*.

workunitid
(input, INT, 4) is a variable containing the work unit to be used to go to a file pool. This must be a signed integer variable with a length of 4. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length5*) contains the information necessary to have CMS return extended error information. If it is specified, it must be followed by a length field.

length5
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable containing a 0 (for synchronous) or a 1 (for asynchronous) processing of the request. The default is 0. On return, *requestid* contains an identifier that is used as input on the DMSCHECK CSL routine to complete the Create External Object request.

Usage Notes

1. See *z/VM: CMS Application Development Guide* for more information about external objects.
2. An external object can be used to store a remote name of an object not managed by the local file pool server. This object can be a file, but does not have to be.
3. External objects can exist only in file control directories.
4. An external object does not have its own authority. Therefore, you cannot grant authority to an external object. If you have write authority to the parent directory, you can create, erase, or rename an external object in that directory. If you have read authority to the parent directory, you can find out if an external object exists with the DMSEXIST CSL routine and retrieve its contents with the

DMSQOBJ CSL routine. An external object can be relocated only by the owner of the directory or by an administrator.

5. If the call to DMSCROB specifies blanks or nothing for *objtype*, DMSQOBJ returns blanks for *objtype*.
6. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the object you are creating.
7. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Create External Object operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
8. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
9. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
10. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to both the *date* and *create_date* parameters.
11. When *date* and *create_date* are specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
12. When *date* and *create_date* are specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the external object.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
13. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSCROB, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSCROB. ERROR means the request failed, or the request was unsuccessful. Errors cause return code 8 or 12.

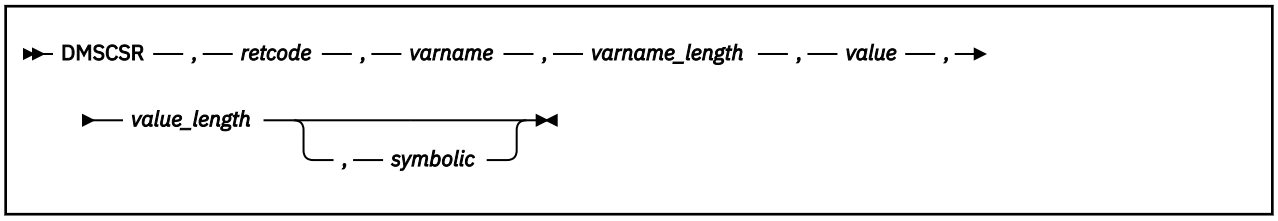
DMSCROB can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	20000	External object or file already exists.
ERROR	44000	Access is not authorized or directory does not exist.

Severity	Reason Code	Description
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	63700	You cannot create an external object in a directory control directory that is accessed as read-only.
ERROR	90250	File name and file type (or *namedef*) are required, but were not specified.
ERROR	90300	Incorrect parameter in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least 1 and not more than 4 tokens in the string.
ERROR	90380	Required parameter omitted in CSL parameter list.
ERROR	90410	Incorrect length specified for one of the character parameters.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name is incorrect. The file name is longer than eight characters or contains an incorrect character.
ERROR	90430	The file type is incorrect. The file type is longer than eight characters or contains an incorrect character.
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the file ID.
ERROR	90472	Incorrect *requestid*, must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Severity	Reason Code	Description
ERROR	90496	Incorrect date specified.
ERROR	90498	Incorrect time format specified.
ERROR	90499	Incorrect time specified.
ERROR	90500	The specified *dirname* is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90540	Specified *workunitid* is incorrect.
ERROR	90590	There is no default file pool currently defined, and file pool ID was not specified as part of the *dirname*.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

DMSCSR - Set a REXX Variable



Context

The format for calling a CSL routine is language dependent. DMSCSR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSCSR
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSCSR" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSCSR routine in a program called from a REXX exec to set the value of a variable in the calling exec. The program calling DMSCSR cannot be written in REXX.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSCSR.

varname
(input, CHAR, *varname_length*) is a variable for specifying the fully-qualified name of the REXX variable.

If the variable name you specify in *varname* has qualifiers, you must specify them. For example, if your variable is the name of an array, you must specify the complete name: the stem followed by a period (`.`), and then the qualifier.

varname_length
(input, INT, 4) is a variable for specifying the length of the REXX variable's name (*varname*).

value
(input, CHAR, *value_length*) is a variable for specifying the new value you want to give to the REXX variable.

value_length
(input, INT, 4) is a variable for specifying the length of the REXX variable's value you are supplying in *value*.

symbolic
(input, INT, 4) is a variable for specifying a flag field. If *symbolic* is equal to zero, REXX searches for the contents of *varname* directly in its variable lists; if *symbolic* is not equal to zero, REXX does normal uppercase translation and substitution for compound variables. This field is optional and defaults to zero, meaning REXX searches for the variable name directly.

For more information about REXX translation and substitution, see the *z/VM: REXX/VM User's Guide*.

Usage Notes

1. You can use this routine to pass values from an application program back to the REXX exec that called the application.

2. When DMSCSR is called from a REXX program, it sets a variable in that program and does not report an error. If the variable is not defined in the program, DMSCSR returns an error code. You can set variables in calling REXX programs by issuing the CMS Pipelines VARS or STEM stage commands from REXX and non-REXX programs (see the *z/VM: CMS Pipelines User's Guide and Reference*.

Return Codes

Note that for the last two return codes listed, the *nn* designates the relative position of the parameter: *retcode* is always parameter number 01, the next parameter is number 02, and so on.

Code
Meaning

0
Normal completion. New value was assigned to existing variable.

112
The number of parameters passed on the call was incorrect.

201
REXX is working and cannot share variables now.

202
REXX is not active.

203
Normal completion. New variable was initialized.

204
Variable name format is invalid.

205
Variable name is too long. (This return code is issued from EXEC2 only.)

207
symbolic parameter not supported. (This return code is issued from EXEC2 only.)

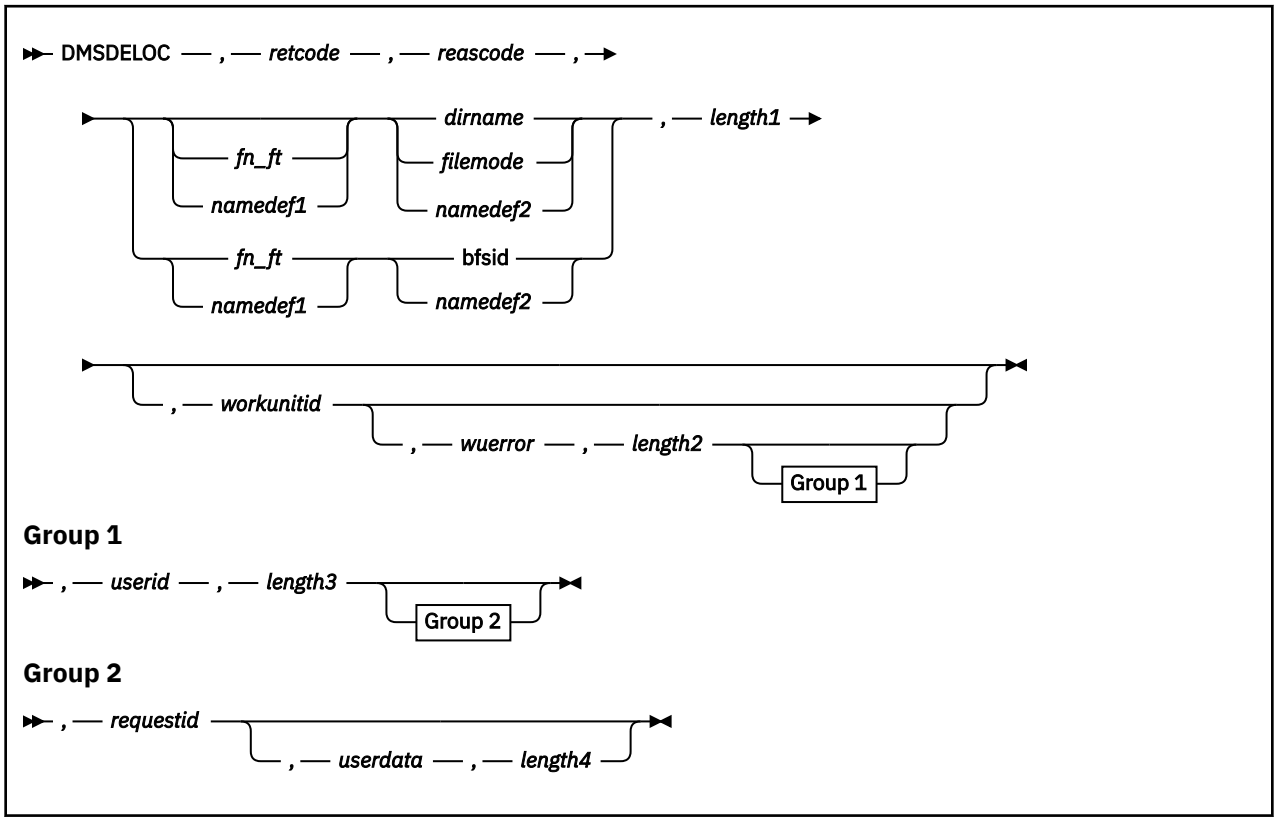
208
Insufficient storage.

209
Storage failure (error in CMSSTOR or SUBPOOL macro).

10*nn*
The data type for parameter *nn* is incorrect.

20*nn*
The length for parameter *nn* is incorrect.

DMSDELOC - Delete Lock



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSDELOC is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSDELOC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSDELOC routine to release an explicit lock created by the DMSCRLOC (Create Lock) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSDELOC.

reascode
(output, INT, 4) is a variable for the reason code from DMSDELOC.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file that is to be unlocked. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the SFS directory name. If *fn_ft* or *namedef1* is specified, this is the directory containing the file that is to be unlocked. If neither *fn_ft* nor *namedef1* is specified, this is the directory that is to be unlocked.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of the accessed SFS directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode. If the file mode letter is not an accessed SFS directory, an error return code will result.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system containing the file that is to be unlocked.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, if specified; plus *dirname*, *filemode*, *bfsid*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

workunitid
(input, INT, 4) is a variable for identifying the work unit associated with the DMSDELOC. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userid
(input, CHAR, 1-8) is the user whose lock is to be released. Only a file pool administrator can use this parameter. If the parameter is omitted (*length3*=0), the requester is assumed to be the holder of the lock.

length3
(input, INT, 4) is a variable containing the length of the preceding character parameter (*userid*). Specifying a length of 0 causes the data in the *userid* parameter to be ignored. This has the effect of omitting the *userid* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

userdata
(input, CHAR, 1-80) is a variable containing a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable containing the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

Usage Notes

1. DMSDELOC provides the function of the CMS DELETE LOCK command, which is described in the *z/VM: CP Planning and Administration* and the *z/VM: CMS Commands and Utilities Reference*.
2. DMSDELOC is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSDELOC causes a noncoordinated commit to be done for the work in the affected file pool.
3. Only the creator of a lock can delete the lock. However, a file pool administrator can use the *userid* parameter to delete a lock set by another user.
4. A user can delete a lock on a file or directory that is still open, as long as the DMSDELOC is from a different work unit.
5. If you specify an alias name, the base file (and its aliases) is unlocked.
6. A request ID is returned if the request is to be asynchronous.
7. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
8. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
9. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file or directory from which you are deleting a lock.

Return Codes and Reason Codes

For lists of the possible return codes from DMSDELOC, see Appendix D, “Return Codes,” on page 597.

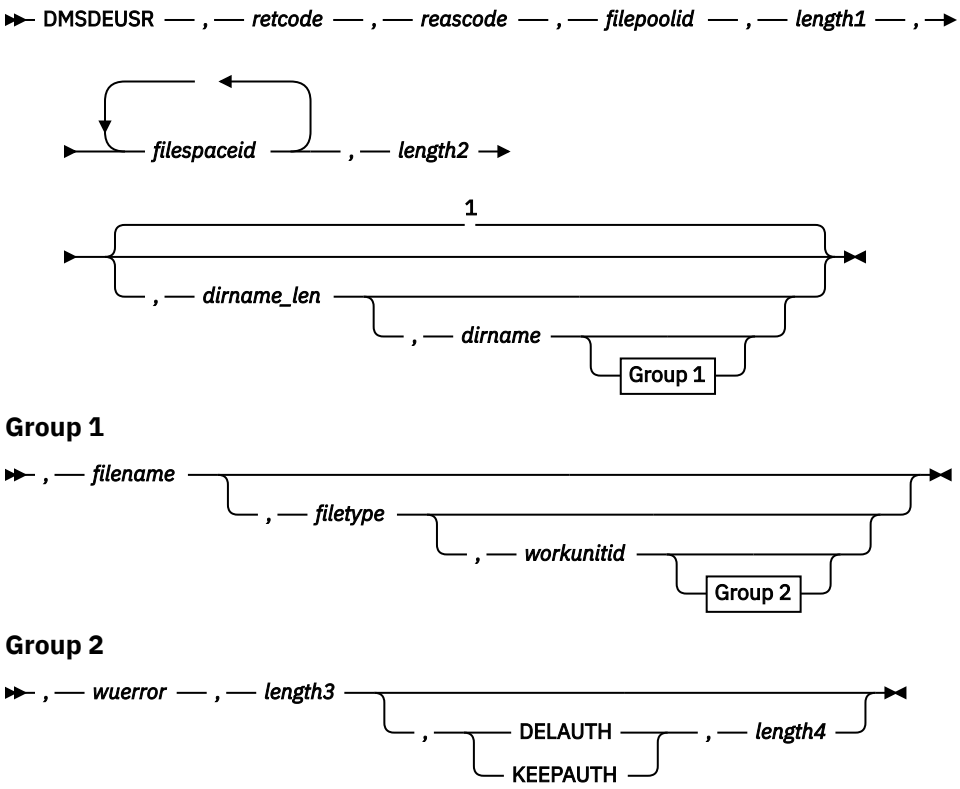
The following table lists the special reason codes returned by DMSDELOC. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSDELOC can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	02050	The specified lock is not held.
ERROR	30000	You do not have the required authority: • In an SFS file space, file pool administration authority is required to delete a lock held by another user. • In a BFS file space, file pool administration authority is required to delete a lock.
ERROR	44000	The specified file pool object does not exist or you are not authorized for it.

Severity	Reason Code	Description
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90310	Invalid option in CSL parameter list. Specified user ID is greater than 8 characters in length.
ERROR	90315	Missing option in CSL parameter list. Specified user ID is all blanks.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.
ERROR	90410	Invalid length specified for *userdata*, file ID, or directory name.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified directory name is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was incorrectly used. For example, a namedef that was created for a directory name was used where a namedef containing a file name and file type expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95400	The specified work unit was already active for the specified file pool when DMSDELOC was executed.

DMSDEUSR - Delete File Space



Notes:

[1] Some optional parameters have nonsyntactical default values; see the parameter descriptions.

Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSDEUSR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSDEUSR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSDEUSR routine to remove one or more file spaces (SFS users or byte file systems) from a file pool. File pool administration authority is required.

Deleting SFS users breaks their connection to the file pool, unless PUBLIC has been enrolled or the users are also administrators.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable to hold the return code from DMSDEUSR.

reascode
(output, INT, 4) is a variable to hold the reason code from DMSDEUSR.

filepoolid
(input, CHAR, 1-8) is a variable that identifies the file pool from which the file space is to be deleted. When specifying this parameter an appended colon is not valid and should not be used.

length1
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter, *filepoolid*.

filespaceid
(input, CHAR, *length2*) is a variable for specifying the file space to be deleted. Specify the length of this variable in *length2*. Nicknames cannot be used. Attempting to delete a file space that is not enrolled results in a warning.

A single file space name can be padded with blanks. In a list, the file space names must be separated by blanks.

In a list, file spaces before a failing file space ID are deleted, but file spaces after the failing ID are not deleted. The user ID index, returned when *wuerror* is specified, identifies the file space ID that caused a failure and any file space IDs that caused warnings.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter, *filespaceid*.

dirname_len
(output, INT, 4) is a variable used to return the length of a directory name. DMSDEUSR returns a value in this parameter if a specified file space cannot be deleted because a file or directory in that file space is locked. In this case *dirname_len* contains the length of the name of the directory that is locked or that contains the file that is locked. The directory name itself is contained in *dirname* (see below).

dirname
(output, CHAR, 153) is a variable used to return the name of the directory that is locked or contains the locked file. The actual length of the directory name is returned in the preceding *dirname_len* parameter.

filename
(output, CHAR, 8) is a variable used to return the name of the file that is locked. If the directory is locked, this parameter is blank.

filetype
(output, CHAR, 8) is a variable in which the file type of the locked file is returned. If the directory is locked, this parameter is blank.

workunitid
(input, INT, 4) is a variable identifying the work unit associated with the DMSDEUSR routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

If the deletion of one of a list of file space IDs fails, the information returned in this parameter identifies the file space ID that caused the failure or the first file space ID that caused a warning.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

DELAUTH
(input, CHAR, 7) deletes all SFS authorizations granted to the file space being removed. If the SFS authorizations status parameter is not specified, DELAUTH is the default.

KEEPAUTH
(input, CHAR, 8) keeps all SFS authorizations granted to the file space being removed.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter.

Usage Notes

1. DMSDEUSR provides the function of the CMS DELETE USER command, which is described in *z/VM: CMS File Pool Planning, Administration, and Operation*.
2. All resources belonging to the file space are freed and returned to the file pool:
   - For an SFS file space (SFS user), all base files, aliases, and directories that the user owned are removed. All authorities that the user granted on the user's files and directories are also removed. Any aliases that other users had on the deleted user's files are erased.
   - For a BFS file space, this includes BFS regular files, directories, special files, and permissions.
3. When DELAUTH is specified or allowed to default, all SFS authorizations granted to the removed file space ID are deleted. If KEEPAUTH is specified, all SFS authorizations granted to the file space ID are kept. Use the KEEPAUTH parameter if you are removing file space data and objects but wish to have the file space ID remain authorized to other objects in the file pool.
4. The Delete File Space routine will fail:
   - If any files or directories in the file space are open or locked, except directories that are open with intent of FILE
   - If there are any uncommitted changes for the deleted SFS user’s files or directories on the work unit being used
   - If one of the deleted file spaces contains a directory control directory that is accessed read/write by anyone other than the administrator issuing the delete request
5. DMSDEUSR is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSDEUSR commits work in the file pool, without coordination.

Return Codes and Reason Codes

For lists of the possible return codes from DMSDEUSR, see Appendix D, “Return Codes,” on page 597.

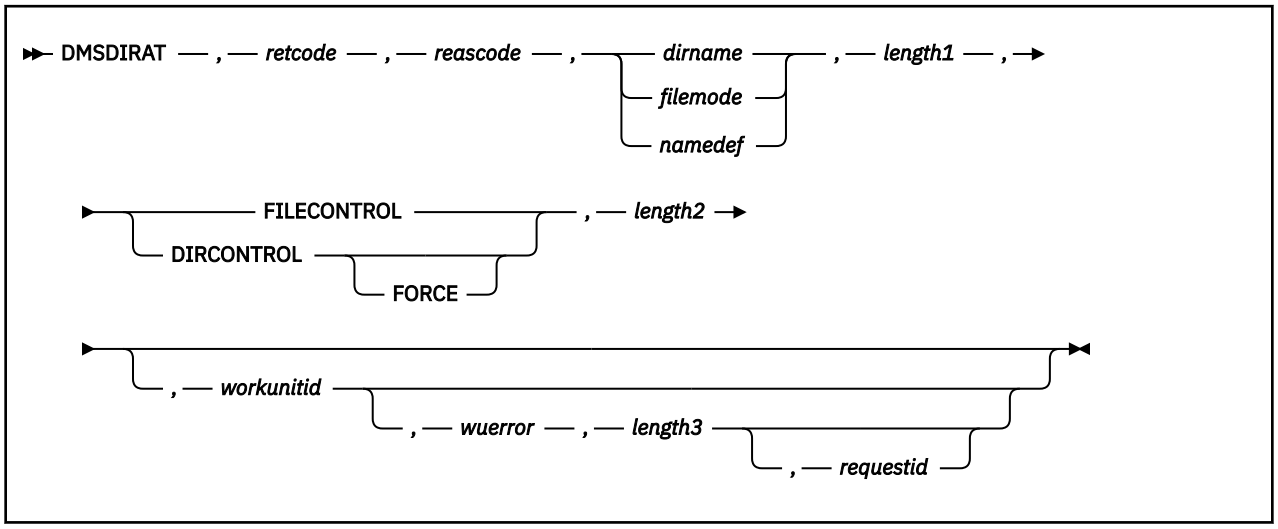
The following table lists the special reason codes returned by DMSDEUSR. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed, or the request was unsuccessful. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSDEUSR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	32040	You attempted to delete a file space that is not enrolled.
WARNING	76010	The request was partially successful. An attempt to delete a file space ID in the list failed after at least one file space ID had been deleted. Use the *userid_index* field of the *wuerror* buffer to identify the failing file space ID.
ERROR	03000	Another user has a directory or file locked.

Severity	Reason Code	Description
ERROR	30000	You do not have the file pool administration authority required to delete a file space.
ERROR	32100	A specified file space cannot be deleted because the file space is currently being accessed.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90210	Extraneous characters in file pool ID specification.
ERROR	90300	Invalid input parameter in CSL parameter list. Specified file space ID is longer than 8 characters. Or, keyword specified preceding *length4* parameter is not valid.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90476	Invalid file pool ID specified.
ERROR	90540	Specified work unit ID is invalid.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.
ERROR	98700	Server is not at a service level that supports the KEEPAUTH parameter.

DMSDIRAT - Set Directory Attribute



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSDIRAT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSDIRAT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSDIRAT routine to set or change the directory attribute of an SFS directory. There are two mutually exclusive values for the directory attribute: file control (FILECONTROL) and directory control (DIRCONTROL). DMSDIRAT lets you switch between these two values.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSDIRAT.

reascode
(output, INT, 4) is a variable for the reason code from DMSDIRAT.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the directory for which you are setting or changing the attribute.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef*) rather than a file mode. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef
(input, CHAR, 1-16) is a variable for specifying a namedef (temporary name) of a directory for which you are setting or changing the attribute.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*dirname* or *namedef*).

DIRCONTROL
(input, CHAR, 10) changes the directory attribute to DIRCONTROL.

FORCE
(input, CHAR, 5) revokes all existing authorities for the directory and its files. FORCE also erases any aliases in the directory.

FILECONTROL
(input, CHAR, 11) changes the directory attribute to FILECONTROL.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (DIRCONTROL and FORCE, or FILECONTROL). See "Compound Variables" on page 15 for coding details.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. DMSDIRAT is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSDIRAT will cause a noncoordinated commit to be done for the work in the affected file pool.
2. For DMSDIRAT to finish successfully when the DIRCONTROL operand is specified, the directory must meet these requirements:
   - Aliases cannot exist in the directory.
   - There are no explicit locks on the files or an explicit lock other than UPDATE on the directory.
   - Authorizations to others cannot exist for files in the directory and to the directory itself (including NEWREAD and NEWWRITE authority).

   While you must delete locks yourself, DMSDIRAT will erase aliases and revoke authorities for you if you specify DIRCONTROL FORCE.
3. Unless you have file pool administration authority, you cannot change the directory attribute of a directory that you do not own.

4. In many cases, you cannot use DMSDIRAT if the directory is currently accessed or in use by you or another user. When you change from file control to directory control, however, specifying the FORCE parameter will make DMSDIRAT succeed so long as there are no open files in the directory.

   Changing from directory control to file control works if you have the directory accessed R/W or others have it accessed R/O. It does not work if you have the directory accessed R/O or someone else has it accessed R/W.

   If you have the directory accessed, it will be released. Other users who have the directory accessed will lose their access along with their authorities.
5. Setting the directory attribute to FILECONTROL revokes the directory's eligibility to be put in a data space.
6. If anyone is currently writing to or reading from the directory or files in the directory, DMSDIRAT will fail and return an error code.
7. When the attribute is changed, any user who has the directory accessed will have it released automatically with no message issued.
8. DMSDIRAT will fail if you are trying to change the attribute to DIRCONTROL on a directory that contains any files that have been migrated by DFSMS/VM. You can recall the files with the DFSMS RECALL command prior to reissuing the request.
9. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
10. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSDIRAT, see Appendix D, “Return Codes,” on page 597.

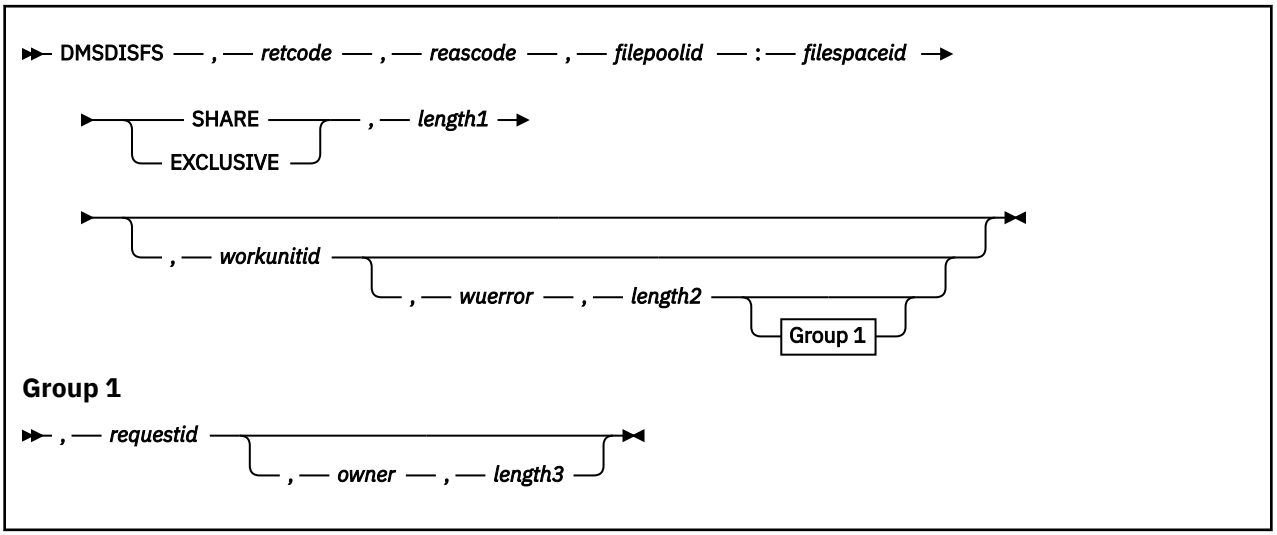
The following table lists the special reason codes returned by DMSDIRAT. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSDIRAT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	61803	The directory attribute was not changed. For example, you specified FILECONTROL for a file control directory.
ERROR	44000	Directory does not exist or you are not authorized.
ERROR	62600	DIRCONTROL parameter specified without also specifying FORCE, and individual authorities exist on the directory or on files within it.
ERROR	62800	Explicit locks are held on the directory or files in the directory.
ERROR	62900	The directory was accessed or in use by an administrator or the owner, when you tried to change the directory attribute from FILECONTROL to DIRCONTROL without specifying FORCE.
ERROR	63000	You could not change the directory attribute from DIRCONTROL to FILECONTROL because either you had the directory accessed R/O or someone else had it accessed R/W.
ERROR	64400	Aliases exist in the directory and you tried to change the directory attribute to DIRCONTROL without specifying FORCE.

Severity	Reason Code	Description
ERROR	65700	The directory contains one or more DFSMS/VM migrated files when you attempted to change the directory attribute to DIRCONTROL.
ERROR	90300	Invalid parameter in CSL parameter list: either 1) the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL; 2) a file mode number was included with the file mode; or 3) a file name or file type was specified as part of the directory name.
ERROR	90320	Conflicting options in CSL parameter list: the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL.
ERROR	90330	Duplicate options in CSL parameter list: the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL.
ERROR	90350	Incorrect number of blank-delimited tokens in the directory name parameter.
ERROR	90380	Missing parameter in CSL parameter list: the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL.
ERROR	90410	Invalid length specified for directory attribute (that is, for DIRCONTROL, FILECONTROL, or DIRCONTROL FORCE).
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The directory name was invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The *namedef* parameter is longer than 16 characters.
ERROR	90530	The namedef does not exist or was used incorrectly. For example a namedef for a directory name was used where a namedef for a file name and file type are expected.
ERROR	90590	There is no default file pool defined and the file pool ID was not specified as part of the directory name.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95400	The specified work unit was already active for the specified file pool when DMSDIRAT was executed.

DMSDISFS - Disable File Space



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSDISFS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSDISFS
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSDISFS routine to place a disable lock on a file space to prevent a particular type of usage by other users.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSDISFS.

reascode
(output, INT, 4) is a variable for the reason code from DMSDISFS.

***filepoolid*:**
(input, CHAR, 2-9) is a variable for specifying the name of the file pool that contains the file space to be disabled. The file pool ID must be followed by a colon.

filespaceid
(input, CHAR, 1-8) is a variable for specifying the name of the file space to be disabled.

SHARE
(input, CHAR, 5) specifies that others can read objects in the file space while you read, but writing objects to the file space is prevented. Several users can have the SHARE lock at the same time. This allows implicit share locks for the same object within the file pool. While a SHARE lock is in effect,

there can be no activity—including DFSMS/VM file migration, recall, and expiration—on objects by other users until the file space is enabled.

When specified, SHARE must be a part of the same character variable that contains *filepoolid*:*filespaceid*. Separate *filepoolid*:*filespaceid* from SHARE with one or more blanks. Specify the length of the entire string in *length1*.

EXCLUSIVE
(input, CHAR, 9) prevents other users from getting any of the locks for this file space. It also prevents users from reading or writing any object in the file space. There can be no activity—including DFSMS/VM file migration, recall, and expiration—on objects by other users until the file space is enabled. Use an Enable File Space (DMSENAFS) routine, the ENABLE operator command, or the FILEPOOL ENABLE administrator command to enable a file space. Files also cannot be put into migrated status, recalled, or erased.

When specified, EXCLUSIVE must be a part of the same character variable that contains *filepoolid*:*filespaceid*. Separate *filepoolid*:*filespaceid* from EXCLUSIVE with one or more blanks. Specify the length of the entire string in *length1*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*filepoolid*:*filespaceid* plus SHARE or EXCLUSIVE).

workunitid
(input, INT, 4) is a variable identifying the work unit associated with DMSDISFS. If you want to specify an optional parameter following *workunitid* without using the work unit ID parameter, specify a value of 0 for *workunitid*.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes error reason code 90415 to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It is not necessary to call DMSCHECK because the function has already been completed.

owner
(input, CHAR, *length3*) is a variable containing the user ID on whose behalf the file space is being disabled and who will own the lock; that is, the request is treated as though it had been issued by *owner* rather than by the actual issuer. Only a file pool administrator may specify this parameter. For an SFS file space, the specified user ID must be either the owner of the file space (in which case, *owner* is identical to the *filespaceid* parameter) or a file pool administrator. For a BFS file space, the specified user ID must be a file pool administrator.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*owner*).

Usage Notes

1. DMSDISFS and DMSDISSG (Disable Storage Group) provide the function of the DISABLE operator command and the FILEPOOL DISABLE administrator command, which are described in the *z/VM: CMS File Pool Planning, Administration, and Operation*.

2. A file space remains disabled until it is enabled. Use the Enable File Space (DMSENAFS) routine, the ENABLE operator command, or the FILEPOOL ENABLE administrator command to enable a file space.
3. DMSDISFS is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it finishes, DMSDISFS causes a noncoordinated commit to be done for the work in the affected file pool.
4. To disable an SFS file space, file space ownership or file pool administration authority is required. To disable a BFS file space, file pool administration authority is required.
5. It is possible for this routine to fail:
   - Because a lock needed to complete the routine is held by some user.
   - Because the file space contains a directory control directory that is accessed read/write by another user. The QUERY ACCESSORS command can be used to determine who has the directory control directory accessed.
6. If the file space is disabled SHARE or EXCLUSIVE, a Disable Storage Group (DMSDISSG) can be done for SHARE or EXCLUSIVE. This will override any file space disable locks. If the storage group is disabled SHARE, a Disable File Space (DMSDISFS) can be done for SHARE. If there is an exclusive lock on a storage group, only the owner of that lock can put a lock on any file space within that storage group.
7. A request ID is returned if the request is to be asynchronous.
8. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
9. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
10. Disabling a BFS file space has the same effect as disabling an SFS file space. For example, if the file space is disabled SHARE, you and others can read objects in the file space, but writing to objects in the file space is prevented (from *any* access method). Similarly, a disable EXCLUSIVE prevents all users, excluding yourself, from reading or writing any object in the file space.

Return Codes and Reason Codes

For lists of the possible return codes from DMSDISFS, see Appendix D, “Return Codes,” on page 597.

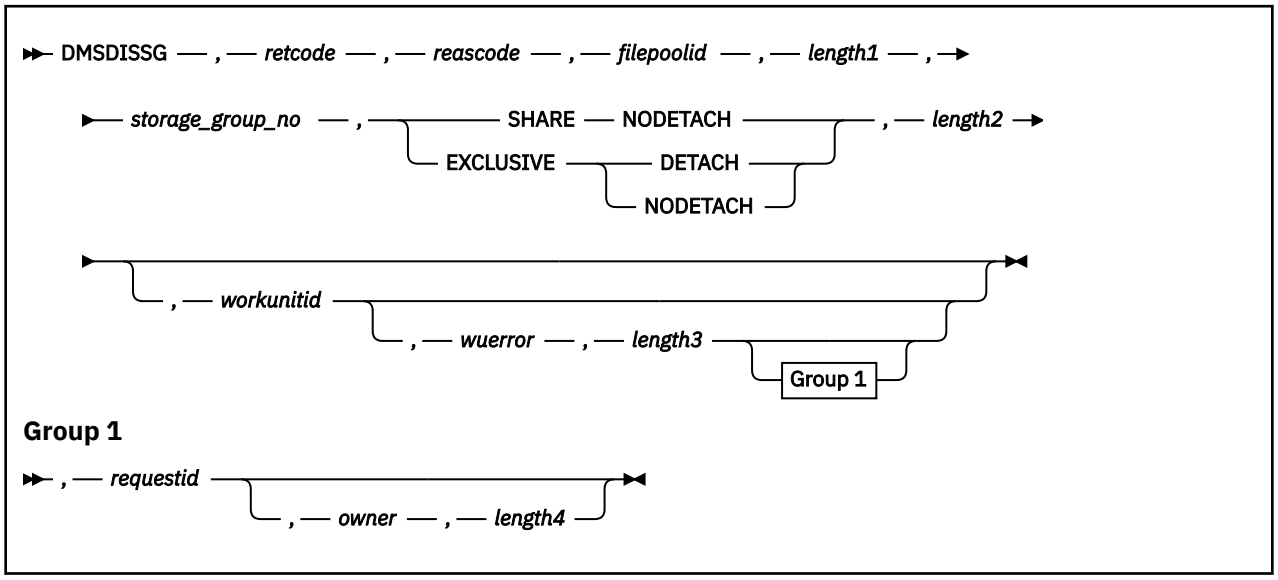
The following table lists the special reason codes returned by DMSDISFS. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR means the request failed, or the request was unsuccessful. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSDISFS can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	02000	You cannot disable the file space because another user has an exclusive lock on the storage group.
WARNING	02080	You already hold the requested lock.
ERROR	02400	You have requested an EXCLUSIVE lock when you or the owner already holds a SHARE lock on the file space.
ERROR	02600	You have requested a SHARE lock when you or the owner already holds an EXCLUSIVE lock on the file space.
ERROR	02700	Another user holds a SHARE lock on the file space and you have requested an EXCLUSIVE lock.

Severity	Reason Code	Description
ERROR	02900	Another user holds an EXCLUSIVE lock on the file space.
ERROR	30000	You do not have the file pool administration authority required to disable a BFS file space.
ERROR	32010	The *filespaceid* part of the *filepoolid*:*filespaceid* parameter is missing or is longer than 8 characters.
ERROR	44000	The file space to be locked does not exist or you are not authorized to it.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90210	Extraneous characters in input parameter.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified. It must be 0 or 1.
ERROR	90476	Incorrect file pool ID specified.
ERROR	90480	Incorrect lock type, must be SHARE or EXCLUSIVE.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

DMSDISSG - Disable Storage Group



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSDISSG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSDISSG
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSDISSG routine to restrict use of a storage group by other users. It places a disable lock on the storage group. File pool administration authority is required.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSDISSG.

reascode
(output, INT, 4) is a variable for the reason code from DMSDISSG.

filepoolid
(input, CHAR, 1-8) identifies the file pool containing the storage group to be disabled. When specifying this parameter, an appended colon is invalid.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*filepoolid*).

storage_group_no
(input, INT, 4) is a number between 2 and 32767 that identifies the storage group within the specified file pool.

SHARE
(input, CHAR, 5) specifies that others may read while you read, but writing objects in the storage group is prevented. Several users can have the SHARE lock at the same time. This allows implicit share locks for the same storage group within the file pool. A SHARE lock prevents DFSMS/VM migration, expiration, and recall of files.

EXCLUSIVE
(input, CHAR, 9) prevents other users from getting either of the locks. No read or write activity is allowed in this storage group by other users until this lock is released. DFSMS/VM file migration, recall and expiration are also not allowed.

DETACH
(input, CHAR, 6) specifies that the file pool server is to detach (CP DETACH command) all minidisks belonging to the storage group so that the caller can link to them. The minidisks are relinked when the storage group is enabled. Once the storage group is disabled with DETACH, DFSMS/VM file migration, recall, and expiration are also not allowed.

DETACH is valid only with EXCLUSIVE.

NODETACH
(input, CHAR, 8) specifies that the file pool server is not to detach any storage group minidisks.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (SHARE or EXCLUSIVE, and DETACH or NODETACH). See "Syntax Conventions for CSL Routines" on page 14 for more information. about coding compound variables.

workunitid
(input, INT, 4) is a variable that identifies the work unit associated with DMSDISSG. If you want to specify an optional parameter following *workunitid* without using the work unit ID parameter, specify a value of 0 for *workunitid*.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the request ID is still 1, no server call was needed and it is not necessary to call DMSCHECK because the function has already been completed.

owner
(input,CHAR,*length4*) is a variable containing the user ID on whose behalf the storage group is being disabled; that is, the request is treated as though it had been issued by *owner* rather than by the actual issuer. The user ID in *owner* should be an administrator's user ID. Otherwise, *owner* will not be able to enable the storage group.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*owner*).

Usage Notes

1. DMSDISSG and DMSDISFS (Disable File Space) provide the function of the DISABLE operator command and the FILEPOOL DISABLE administrator command which are described in the *z/VM: CMS File Pool Planning, Administration, and Operation*.

2. The storage group remains disabled until it is enabled. Use the Enable Storage Group (DMSENASG) routine, the ENABLE operator command, or the FILEPOOL ENABLE administrator command to enable the storage group.
3. DMSDISSG is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSDISSG commits the work without coordination.
4. This routine fails if:
   - A lock needed to complete the routine is held by some user.
   - The storage group contains a directory control directory that is accessed read/write by another user. Other users accessing the directory in read-only mode lose access without any message. The QUERY ACCESSORS command can be used to determine who has the directory control directory accessed.
5. If the file space in the storage group is disabled SHARE or EXCLUSIVE, a Disable Storage Group (DMSDISSG) can be done for SHARE or EXCLUSIVE. This will override any file space disable locks. If the storage group is disabled SHARE, a Disable File Space (DMSDISFS) can be done for SHARE. If the storage group is disabled EXCLUSIVE, a Disable File Space (DMSDISFS) cannot be done.
6. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
7. A request ID is returned if the request is to be asynchronous.
8. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSDISSG, see Appendix D, “Return Codes,” on page 597.

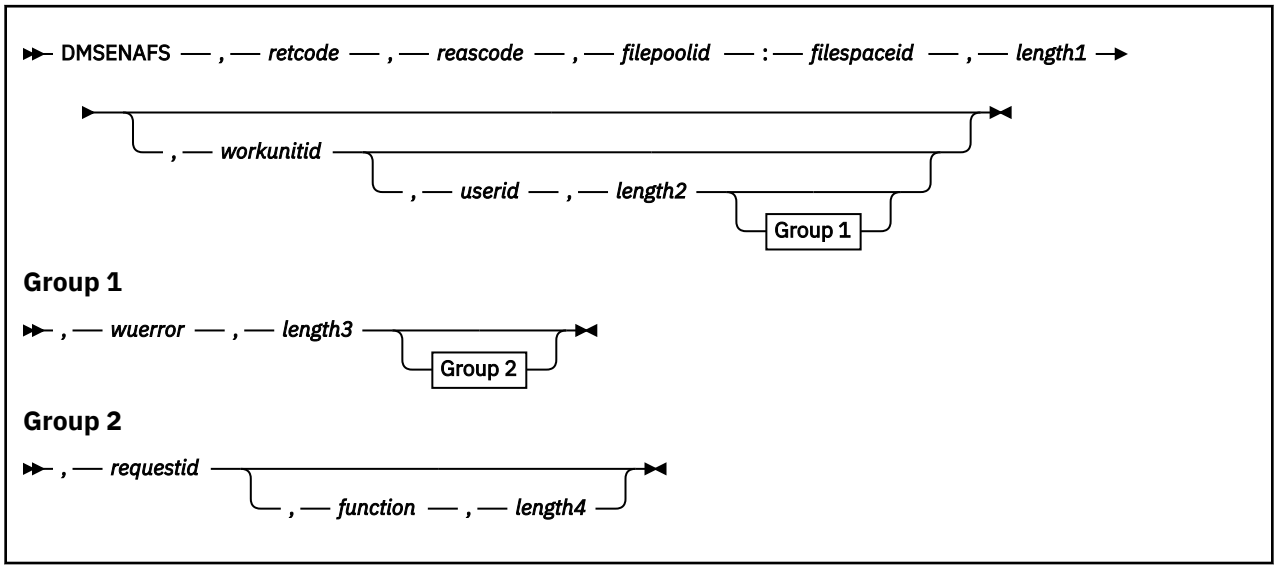
The following table lists the special reason codes returned by DMSDISSG. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSDISSG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	02080	You already hold the requested lock.
ERROR	02400	You have requested an EXCLUSIVE lock when you or the owner already holds a SHARE lock on the storage group.
ERROR	02600	You have requested a SHARE lock when you or the owner already holds an EXCLUSIVE lock on the storage group.
ERROR	02700	Another user holds a SHARE lock on the storage group for which you have requested an EXCLUSIVE lock.
ERROR	02900	Another user holds an EXCLUSIVE lock on the storage group.
ERROR	30000	You are not authorized to issue this request or the *owner* specified is not authorized to perform this request.
ERROR	50100	Specified storage group number is invalid (less than 2 or greater than MAXDISKS server parameter).
ERROR	50200	Specified storage group does not exist.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.

Severity	Reason Code	Description
ERROR	90210	Extraneous characters in one of the following parameters: SHARE|EXCLUSIVE, or DETACH|NODETACH.
ERROR	90320	Conflicting options. DETACH was specified and the lock type was not EXCLUSIVE.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified, must be 0 or 1.
ERROR	90476	Invalid file pool ID specified.
ERROR	90478	Invalid parameter, must be DETACH or NODETACH.
ERROR	90480	Invalid lock type, must be SHARE or EXCLUSIVE.
ERROR	90540	Specified work unit ID is invalid.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

DMSENAFS - Enable File Space



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSENAFS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSENAFS
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSENAFS routine to release a disable lock on a file space that was set using the DMSDISFS (Disable File Space) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSENAFS.

reascode
(output, INT, 4) is a variable for the reason code from DMSENAFS.

***filepoolid*:**
(input, CHAR, 2-9) is a variable for specifying the name of the file pool that contains the file space to be enabled. The file pool ID must be followed by a colon.

filespaceid
(input, CHAR, 1-8) is a variable for specifying the name of the file space to be enabled.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*filepoolid*:*filespaceid*).

workunitid
(input, INT, 4) is a variable that identifies the work unit to be associated with DMSENAFS. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

userid
(input, CHAR, 1-8) is a variable for specifying the user ID whose disable lock is to be removed, if the user of the routine is not the one who disabled the file space. Only a file pool administrator may specify this parameter.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userid*). Specifying a length of 0 causes the data in the *userid* parameter to be ignored. This has the effect of omitting the *userid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

function
(input, CHAR, *length4*) is a variable identifying the function that disabled the file space and its storage group. If this parameter is specified, the *userid* parameter must not be specified. The list of valid functions currently consists of one name, 'RENAME'.

length4
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*function*).

Usage Notes

1. DMSENAFS and DMSENASG (Enable Storage Group) provide the function of the ENABLE command, which is described in *z/VM: CMS File Pool Planning, Administration, and Operation*.
2. DMSENAFS is an atomic request. This means that there can be no outstanding work for the affected file pool on the work unit when this routine is called. When it is finished, DMSENAFS commits the work without coordination.
3. Only the user who disabled the file space—using the Disable File Space routine (DMSDISFS), the FILEPOOL DISABLE administrator command, or the DISABLE operator command—can execute the Enable File Space routine. An exception is that a file pool administrator can use the routine, specifying the user ID of the person who disabled the file space in the *userid* parameter.
4. A request ID is returned if the request is to be asynchronous.
5. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
6. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

7. When the *function* parameter is used, the specified file space and its associated storage group, which were disabled due to an SFS rename user ID attempt using the FILEPOOL RENAME command, are unlocked.
8. This routine fails if:
   - A lock needed to complete the routine is held by some user.
   - The file space contains a directory control directory that is accessed read/write.

Return Codes and Reason Codes

For lists of the possible return codes from DMSENAFS, see Appendix D, "Return Codes," on page 597.

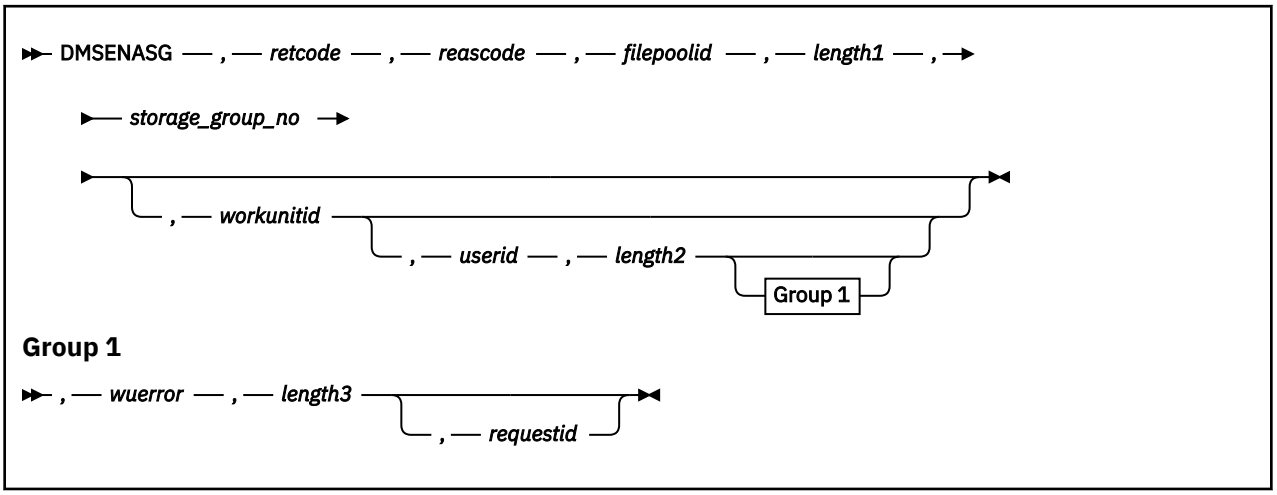
The following table lists the special reason codes returned by DMSENAFS. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSENAFS can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	02050	The specified lock is not held.
WARNING	02070	There are no locks held on the object by *function* RENAME.
ERROR	30000	You do not have the required authority: • To enable an SFS file space that was disabled by another user, you must have file pool administration authority. • To enable a BFS file space, you must have file pool administration authority.
ERROR	32010	The *filespaceid* part of the *filepoolid*:*filespaceid* parameter is missing or is longer than 8 characters, or the *userid* parameter is longer than 8 characters.
ERROR	44000	The file space does not exist or you are not authorized to enable it.
ERROR	67000	Function specified is incorrect (currently the only valid function is RENAME).
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90210	Extraneous characters in an input parameter.
ERROR	90320	Specifying both *function* and *userid* parameters is not allowed.
ERROR	90410	Incorrect parameter length specified. The name in *function* has a length greater than 8.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified, must be 0 or 1.
ERROR	90476	Incorrect file pool ID specified.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

Severity	Reason Code	Description
ERROR	98700	File pool is at a release level that does not support enable for a *function* request.

DMSENASG - Enable Storage Group



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSENASG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSENASG
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSENASG routine to release a disable lock on a storage group that was set using the DMSDISSG (Disable Storage Group) routine. File pool administration authority is required.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSENASG.

reascode
(output, INT, 4) is a variable for the reason code from DMSENASG.

filepoolid
(input, CHAR 1-8) is a variable that identifies the file pool containing the storage group to be enabled. When specifying this parameter, an appended colon is incorrect and should not be used.

length1
(input, INT, 4) is a variable containing the length of the preceding character parameter (*filepoolid*).

storage_group_no
(input, INT, 4) is a variable containing a fullword number between 2 and 32767, which identifies the storage group within the specified file pool.

workunitid
(input, INT, 4) is a variable identifying the work unit associated with the DMSENASG routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for *workunitid*.

userid
(input, CHAR, 1-8) is the variable identifying the user for whom the enable is being done. If not specified, it defaults to the user ID of the issuing administrator machine. When specifying this parameter, note that nicknames are not allowed and should not be used.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userid*). Specifying a length of 0 causes the data in the *userid* parameter to be ignored. This has the effect of omitting the *userid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. It is a signed integer variable, with a length of 4. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It is not necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. DMSENASG and DMSENAFS (Enable File Space) provide the function of the ENABLE command, which is described in the *z/VM: CP Planning and Administration*.
2. Only the user who disabled the storage group (using the Disable Storage Group routine or the DISABLE operator command) can enable it, except that a file pool administrator can use the routine, specifying the user ID of the person who disabled the storage group.
3. If DETACH is specified on the Disable Storage Group (DMSDISSG), a detach must be done from this virtual machine prior to entering the Enable Storage Group. The file pool server relinks those minidisks at this time. The enable fails if it is unable to successfully link.
4. DMSENASG is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it finishes, DMSENASG causes a noncoordinated commit to be done for the work in the affected file pool.
5. This routine fails if any of these conditions exist:
   - A lock needed to complete the routine is held by some user.
   - The storage group contains a directory control directory that is accessed read/write.
   - You are using data spaces, and the storage group contains an FBA disk that is not allocated in 8-block increments or that is not aligned on a 4K boundary.
6. A request ID is returned if the request is to be asynchronous.
7. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.

8. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

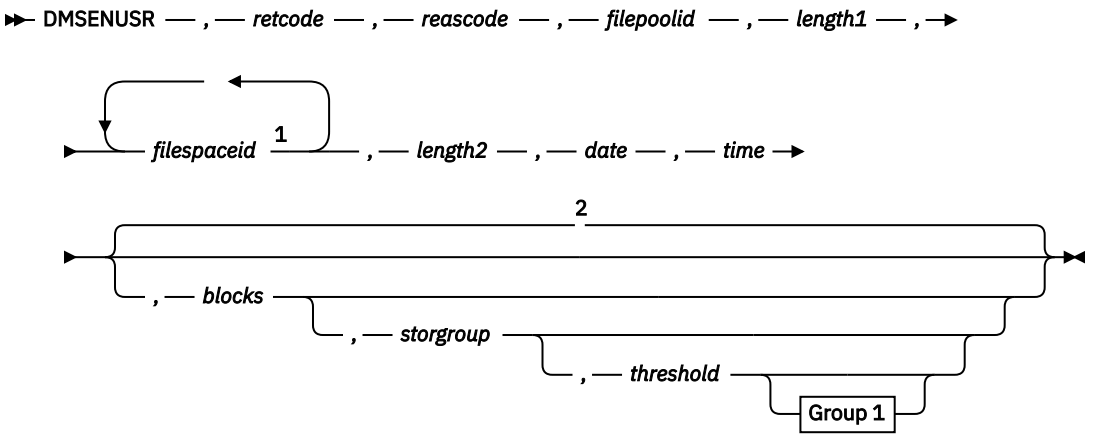
For lists of the possible return codes from DMSENASG, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSENASG. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

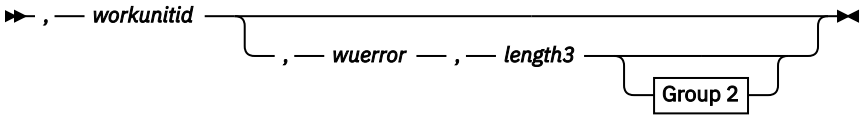
DMSENASG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	02050	The specified lock is not held.
ERROR	30000	You are not a file pool administrator, and the request is to enable a storage group that was disabled by another user.
ERROR	32010	The user ID parameter is longer than 8 characters.
ERROR	44000	The storage group does not exist or you are not authorized to enable it.
ERROR	50100	Incorrect storage group specified.
ERROR	66400	The storage group was not be enabled, because it contains an FBA minidisk that is not aligned on a 4K boundary. This restriction applies only if you have created data spaces since the file pool server started up.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect requestid specified, must be 0 or 1.
ERROR	90476	Incorrect file pool ID specified.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

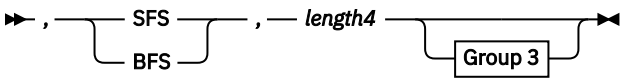
DMSENUSR - Enroll File Space



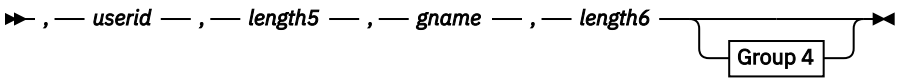
Group 1



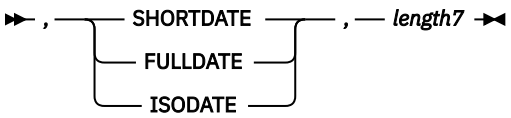
Group 2



Group 3



Group 4



Notes:

[1] Multiple file space IDs may be specified only for SFS.
[2] Some optional parameters have nonsyntactical default values; see the parameter descriptions.

Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSENUSR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSENUSR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSENUSR routine to enroll one or more SFS users or one byte file system in a file pool. File pool administration authority is required.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSENUSR.

reascode
(output, INT, 4) is a variable for the reason code from DMSENUSR.

filepoolid
(input, CHAR, 1-8) is a variable that identifies the file pool in which the file space is to be enrolled. When specifying this parameter, an appended colon is not valid and should not be used.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*filepoolid*).

filespaceid
(input, CHAR, *length2*) is a variable for specifying the name of the file space to be created. For SFS, several names may be specified, separated by blanks. For BFS, only one name may be specified. Nicknames are not allowed for this parameter.

Do not enroll a name that begins with a plus (+) or a minus (-) or that contains a colon (:) or a period (.). These characters are used as separator characters in SFS directory IDs, of which the file space ID is a part. Also, BFS file space names may not contain the slash (/) or null (X'00') characters.

Because DFSMS/VM uses user IDs beginning with *DFSMS*, you should avoid enrolling a name that begins with *DFSMS*. If you do enroll such a name, DFSMS/VM will not manage anything in that file space. See *z/VM: DFSMS/VM Storage Administration* for more information.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (*filespaceid*).

date
(input, CHAR, 8 or 10) is the date when the top directory was created. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. The date is in local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the time used for when the top directory was created. Specify the time in the form *hh:mm:ss*, where *hh* is the hour of the day in 24-hour notation, *mm* is the minutes and *ss* is the seconds. If this field is set to 8 blanks, then the current time is used.

blocks
(input, INT, 4) is a variable specifying the upper limit of file blocks (4KB blocks) the file space being enrolled can use.

If the number of blocks is not specified or is 0, then the file space is given authority to connect to the file pool and is given a top directory. If it is an SFS file space, the owner of the file space (*filespaceid*) or a file pool administrator can create a directory structure and put aliases in it, but cannot create base files in the directory structure. If it is a BFS file space, any BFS user with the appropriate permission can create all BFS objects in the file space except regular files. For either type of file space, you can later use the MODIFY USER command to allocate space to it.

storgroup
(input, INT, 4) is a variable for specifying the storage group to which the file space will be assigned. This parameter must be a number between 2 and 32767 and not be greater than the MAXDISKS value for the file pool. If not specified, this parameter value defaults to 2.

threshold
(input, INT, 4) is a variable indicating the file space warning threshold percentage. For an SFS file space, this must be a value between 1 and 99. If not specified, this parameter value defaults to 90. For a BFS file space, the warning threshold is ignored; the value is always 100.

workunitid
(input, INT, 4) is a variable identifying the work unit associated with the DMSENUSR routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

If a list of SFS users is specified in the *filespaceid* parameter and enrolling a user fails, the information returned when this parameter is used will identify the user ID which caused the failure or the first user ID which caused a warning.

SFS
(input, CHAR, 3) means the enrolled file space is an SFS user. If the file space type parameter is not specified, SFS is the default.

BFS
(input, CHAR, 3) means the enrolled file space is a byte file system.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter. Specifying 0 has the same effect as omitting the preceding parameter.

userid
(input, CHAR, 1-8) is a variable for specifying the VM user ID whose POSIX user ID (UID) should become the owning UID of the top directory of the BFS file space. Nicknames are not allowed. This parameter is not valid for SFS file spaces.

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter. Specifying 0 has the same effect as omitting the *userid* parameter.

gname
(input, CHAR, 1-8) is a variable for specifying the POSIX group name whose associated POSIX group ID (GID) should become the owning GID of the top directory of the BFS file space. This parameter is not valid for SFS file spaces.

length6
(input, INT, 4) is a variable for specifying the length of the preceding character parameter. Specifying 0 has the same effect as omitting the *gname* parameter.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. If the date format parameter is not specified, SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length7
(input, INT, 4) is a variable containing the length of the preceding character parameter (SHORTDATE, FULLDATE, or ISODATE).

Usage Notes

1. DMSENUSR provides the function of the ENROLL USER command, which is described in *z/VM: CMS File Pool Planning, Administration, and Operation*.
2. You can change an enrolled user's storage group through a combination of the FILEPOOL UNLOAD and FILEPOOL RELOAD FILESPACE commands.
3. DMSENUSR is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSENUSR causes a noncoordinated commit to be done for the work in the affected file pool.
4. If you wish to allow anyone (public) to connect to a file pool, use the ENROLL PUBLIC command. Specifying 'public' as *filespaceid* in a call to DMSENUSR allows a specific user ID 'public' to connect to the file pool; it does **not** allow all users to connect to the file pool.
5. When enrolling a byte file system, the user ID and group name specified on the *userid* and *gname* parameters are translated into the UID and GID values obtained from the POSIX user database. If you do not specify the user ID and group name on this routine, the values are set to the current effective UID and GID. The POSIX database can be in the z/VM CP directory or managed by an external security manager (ESM). Note that the UID and GID values are obtained from the database on the z/VM system on which this routine is issued. Therefore, the administrator issuing this routine should be on the same z/VM system as the file pool into which the byte file system is being enrolled, or results are unpredictable.
6. When creating a BFS file space, the requesting VM user ID can specify an owning UID (*userid*) only if authorized to obtain a user entry in the POSIX database. The requestor can specify an owning GID (*gname*) only if authorized to obtain a group entry in the database.

   Typically, the requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

   If that is not the case, one of the following must be true:

   - The external security manager (ESM) grants the requestor the authority to read the entry.
   - An ESM is either not installed or defers authorization to CP, and one of the following is true:
     - The requestor is a superuser (has an effective UID of 0).
     - The current real or effective UID or GID matches the UID or GID for the specified user or group.
     - For a GID, the requestor is a member of the specified group.

7. A BFS file space contains a byte file system and is typically accessed by many users. In addition to creating the BFS file space, you may also need to create individual file spaces for those users who will be accessing BFS data. These file spaces can be either SFS or BFS. You may want to specify 0 as the *blocks* parameter, because the users will be using space in the BFS file space, not SFS. Or, you may want to consider using the ENROLL PUBLIC command to allow public access to your file pool.
8. If the file space is a BFS file space and the date provided is earlier than January 1, 1970, the time and date of creation of the top directory are set to 00:00:00 and January 1, 1970.
9. When a BFS file space is created, the owning UID is given read, write, and search permission for the top directory. No group or public permissions are given. The owner of the file space or a BFS superuser can use the OPENVM PERMIT command to change permissions associated with the directory.
10. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
11. When *date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
12. When *date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the creation of the top directory.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
13. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSENUSR, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSENUSR. ERROR indicates that the request failed. Errors cause return code 8 or 12.

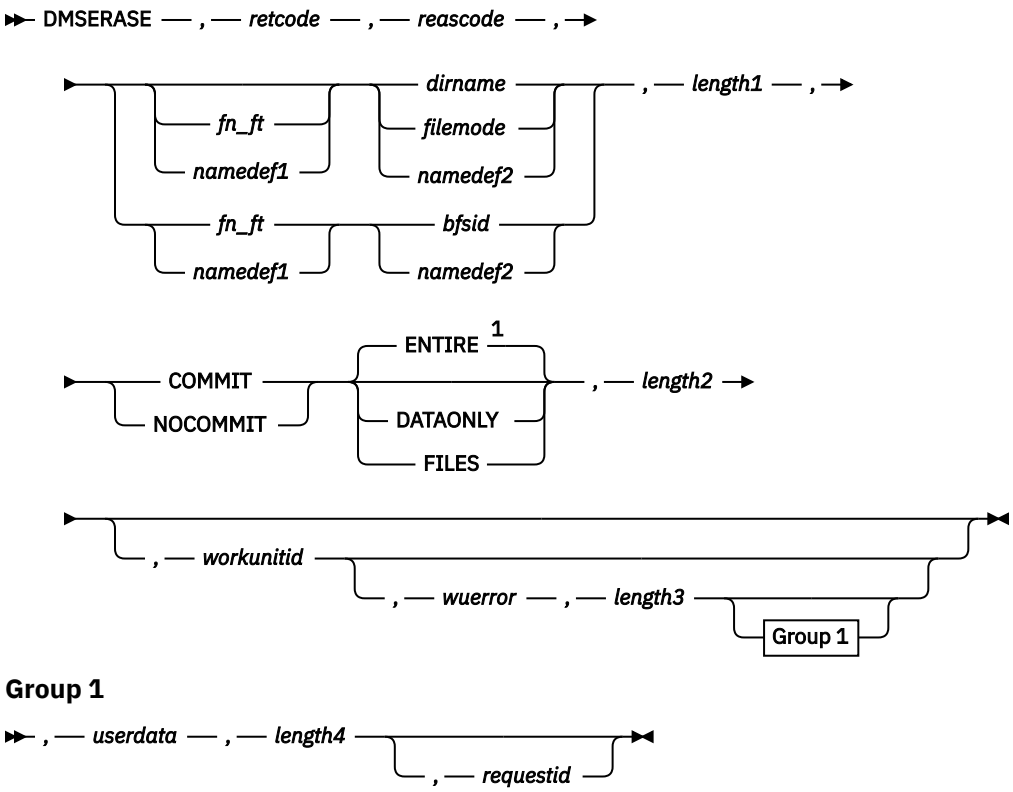
DMSENUSR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	10210	Error obtaining UID or GID
ERROR	10211	Error obtaining UID or GID. Insufficient storage
ERROR	10212	Error obtaining UID or GID. User not authorized
ERROR	10213	Error obtaining UID or GID. CMS internal error
ERROR	10214	Error obtaining UID or GID. CP internal error
ERROR	10215	Error obtaining UID or GID. Database not available
ERROR	10216	Error obtaining UID or GID. User not found
ERROR	10217	Error obtaining UID or GID. Group not found

Severity	Reason Code	Description
ERROR	10240	Function not allowed in DOS mode, in SUBSET mode, or on this level of CP.
ERROR	20000	A specified file space ID is already enrolled.
ERROR	30000	You do not have the file pool administration authority required to enroll a file space.
ERROR	50100	Specified storage group number is invalid (less than 1 or greater than MAXDISKS server parameter).
ERROR	50200	Specified storage group does not exist.
ERROR	50300	Specified storage group does not exist and MAXDISKS limit reached by file pool server.
ERROR	50400	You cannot assign a file space to storage group 1.
ERROR	51055	Specified warning threshold percentage is invalid.
ERROR	69000	Only 1 file space ID can be specified if the file space type is BFS.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90210	Extraneous characters in file pool ID specification.
ERROR	90300	Invalid input parameter in CSL parameter list. For example, a specified SFS or BFS file space ID is longer than 8 characters, begins with a plus (+) or minus (-), or contains a colon (:) or period (.), or a specified BFS file space ID contains a slash (/) or null character (X'00').
ERROR	90310	Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, or ISODATE.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90476	Invalid file pool ID specified.
ERROR	90490	Invalid number of file space blocks specified.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Nonnumeric value in date specification.

Severity	Reason Code	Description
ERROR	90498	Invalid time format; must be in the form HH:MM:SS.
ERROR	90499	Nonnumeric value in time specification.
ERROR	90540	Specified work unit ID is invalid.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.
ERROR	98700	Server is not at a service level that supports BFS file spaces.

DMSERASE - Erase



Group 1

Notes:

[1] ENTIRE is the default only when *fn_ft* or *namedef1* is specified.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSERASE is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSERASE
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSERASE routine to erase one of the following:

- Minidisk file
- SFS file, alias, external object, or directory
- BFS regular file

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSERASE.

reascode
(output, INT, 4) is a variable for the reason code from DMSERASE.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file, alias, or external object that is to be erased. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the SFS directory name. If *fn_ft* or *namedef1* is specified, this is the directory containing the file, alias, or external object that is to be erased. If neither *fn_ft* nor *namedef1* is specified, this is the directory that is to be erased.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of the minidisk file, SFS file, or SFS directory that is to be erased. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode. If the file mode is not an accessed minidisk or SFS directory, an error return code will result.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system containing the file that is to be erased.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, if specified; plus *dirname*, *filemode*, *bfsid*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

COMMIT
(input, CHAR, 6) means keep all changes made during a work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

DMSERASE of a BFS file is committed even if NOCOMMIT is specified. However, this does not affect other participating resources updated on the work unit, including SFS files.

ENTIRE
(input, CHAR, 6) means to erase all information about a file, alias, or external object.

For an SFS base file or a minidisk file, all file data, catalog information, and control information relating to the file is removed. All authorizations granted to the file are lost.

When the specified file ID is an alias, the alias is deleted from the directory, but the base file is not affected.

For a BFS file, all file data, catalog information, and control information related to the file is removed. All hard links are erased.

ENTIRE is the default when *fn_ft* or *namedef1* is specified.

ENTIRE is not valid for directories.

DATAONLY
(input, CHAR, 8) means to erase the contents of a file. After the erasure, only an empty file with the specified name remains.

For an SFS file, catalog information is retained along with control information, existing file authorizations, and aliases. When DATAONLY is specified for an alias, the contents of the base file are deleted. The alias remains.

For a BFS file, catalog information is retained along with control information, existing file permission settings, and links.

DATAONLY is not valid for:

- Directories
- External objects
- Minidisk files

FILES
(input, CHAR, 5) means to erase an SFS directory and all the files, aliases, and external objects it contains. However, if subdirectories are present, nothing is erased.

Because you cannot erase a BFS top directory using this routine, the FILES parameter is not valid for BFS.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (COMMIT or NOCOMMIT; plus ENTIRE, DATAONLY, or FILES, if specified). See "Compound Variables" on page 15 for coding details.

workunitid
(input, INT, 4) is a variable for identifying the work unit associated with the DMSERASE routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see usage note "2" on page 155).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

Common Notes for SFS, BFS, and Minidisk

1. If you call DMSERASE for a nonrecoverable file and then do a rollback, the deletion generally still becomes permanent.
2. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the object you are erasing. If the directory or file being erased is on a minidisk, *userdata* is not used.
3. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Erase operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
4. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Notes for SFS Only

1. When DMSERASE erases a base file with the ENTIRE option, it removes all aliases to and authorizations for a file. Once this operation is committed, aliases of the file are considered to be erased aliases.
2. You can erase another user's alias with either ENTIRE or DATAONLY specified. With the ENTIRE option, only the alias is deleted. If you specify DATAONLY, the contents of the file are deleted.
3. The authorities that you have to an SFS file and directory affect whether you can use the ENTIRE and DATAONLY options to erase another user's SFS files and aliases. Table 18 on page 155 summarizes the interactions for both files and aliases.
4. A directory can be erased only by the directory owner or the file pool administrator. DMSERASE fails if the owner has opened the directory with DMSOPDIR. The administrator can erase a directory even if the owner has it open, but not if the administrator has it open. However, if another user has opened the directory with DMSOPDIR, the directory owner or an administrator can erase the directory.

Table 18. Erasing SFS Files and Directories. Using the DMSERASE Routine with ENTIRE and DATAONLY

Call DMSERASE against	Option	Necessary authorities			Results
		File write	Directory write	File read	
File in file control directory	ENTIRE	•	•		File and all related authorizations, aliases, and control data are erased.
File in file control directory	DATAONLY	•			Only contents of file are deleted; aliases, authorizations, control data, and empty file remain.
Alias	ENTIRE		•	•	Alias is deleted; base file is unaffected.
Alias	DATAONLY	•			Contents of base file are deleted.
File in directory control directory	ENTIRE		•		File and all related authorizations, aliases, and control data are erased.

Table 18. Erasing SFS Files and Directories. Using the DMSERASE Routine with ENTIRE and DATAONLY *(continued)*

Call DMSERASE against	Option	Necessary authorities			Results
		File write	Directory write	File read	
File in directory control directory	DATAONLY		•		Only contents of file are deleted; aliases, authorizations, control data, and empty file remain.

5. A file or directory cannot be erased under the following conditions:
   - The directory was opened with DMSOPDIR. However, if the directory is open with an intent of FILE, it can be erased. You can also erase files in a directory that is open.
   - The file is open in read/write mode by any user or open in read-only mode by you; it can be open in read-only mode by others.
   - The file or directory is locked by another user.
   - Any user (including the issuer of DMSERASE) has a SHARE lock on the file or directory.

   If the issuer has an UPDATE or EXCLUSIVE lock on the file or directory, the erase request is allowed.
6. If any files within the directory to be erased meet the criteria stated in the previous usage note, the directory cannot be erased.

   You cannot erase a directory control directory, or a file or subdirectory in a directory control directory, if:
   - You have the parent directory accessed read-only
   - Any other user has the directory accessed read/write.
7. When entered for a directory, DMSERASE fails if the directory contains subdirectories, if any files are locked, or if any files in the directory are open. If FILES is not specified and the directory contains files other than dropped or revoked aliases, (such as base files, aliases, or external objects) the command fails.
8. When a directory is erased, the recoverability attribute of the files is ignored. This means that if the erase request is rolled back, no file in the directory is erased.
9. If a directory is erased within an active work unit, any accesses for a directory by that name are released when the work is committed
10. If a directory is erased within an active work unit and later a directory by the same name is created within the work unit, when the work is committed, if the directory is open with intent FILE, it is automatically closed.
11. When you issue DMSERASE for an external object, the external object is erased. The erasure has no effect on the object designated by the remote name in the external object.

Notes for BFS Only

1. You cannot erase a BFS file if:
   - The file is not a BFS regular file.
   - The file is open.
   - The file is locked by another user.
   - Any user (including the issuer of DMSERASE) has a SHARE lock on the file.
   - You are not a file pool administrator.
2. DMSERASE of a BFS file is committed whether or not COMMIT is specified. This does not affect other participating resources updated on the work unit, including SFS files. Changes made to BFS files are not protected. That is, they do not participate in CRR and are committed independently of other resources, including SFS, on the work unit.

Notes for Minidisk Only

1. A minidisk file cannot be erased if it is already opened.
2. Minidisk files are erased permanently as part of DMSERASE processing. If the work unit is rolled back or there is a system failure, the file is not restored.
3. Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Return Codes and Reason Codes

For lists of the possible return codes from DMSERASE, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSERASE. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

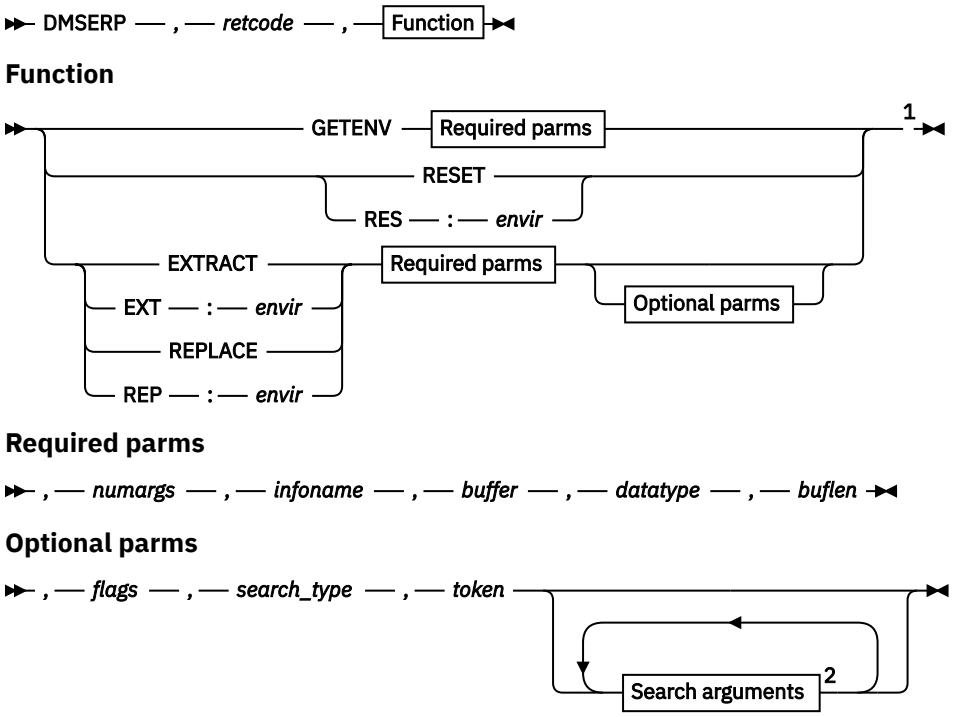
DMSERASE can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51050	Still exceeding file space warning threshold after erasing the file or directory.
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit processing.
WARNING	90351	DATAONLY option was specified and file is already empty.
ERROR	03100	The specified file or directory, or a file in the specified directory is explicitly locked by the requester.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	44000	The specified directory or file does not exist or you are not authorized to erase it.
ERROR	44100	You are not authorized to erase one or more objects in the directory.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.

Severity	Reason Code	Description
ERROR	51100	COMMIT was specified. System error. No minidisks assigned to the storage group.
ERROR	60000	The specified directory contains files, aliases, or external objects, and the FILES option was not specified.
ERROR	60100	You have the specified file or one or more files in the specified directory open.
ERROR	60200	You cannot erase a top-level directory.
ERROR	60300	Specified directory contains one or more subdirectories.
ERROR	60400	You have the directory open.
ERROR	63700	Specified directory control directory is accessed read-only.
ERROR	65400	DATAONLY is invalid for an external object, or you tried to erase a BFS object that is not a regular file.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT option was specified.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90220	The specified file does not exist or you are not authorized to erase it.
ERROR	90230	The specified directory does not exist or you are not authorized to it.
ERROR	90305	FILES option is invalid when you are erasing a file.
ERROR	90308	Specified option is invalid for directory.
ERROR	90310	Invalid parameter in CSL parameter list: COMMIT, NOCOMMIT, DATAONLY, ENTIRE, or FILES not specified correctly.
ERROR	90315	Missing parameter in CSL parameter list: COMMIT or NOCOMMIT not specified.
ERROR	90320	Conflicting parameter in CSL parameter list: COMMIT and NOCOMMIT, or DATAONLY and ENTIRE, were specified.
ERROR	90330	Duplicate parameter in CSL parameter list: COMMIT, NOCOMMIT, DATAONLY, ENTIRE, or FILES already specified.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than three tokens in the string.
ERROR	90410	Invalid length specified for *userdata*, COMMIT, NOCOMMIT, DATAONLY, ENTIRE, or FILES parameter.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than eight characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than eight characters or contains an invalid character.

Severity	Reason Code	Description
ERROR	90450	Global file name characters (* or %) were found in the *fileid* parameter.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified directory name is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The *namedef* part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The *namedef* part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a temporary name (*namedef*) that was created for a directory name was used where a temporary name for a file was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Provided only a file mode letter as input but it corresponds to a minidisk. An option or parameter specified is invalid with a minidisk.
ERROR	90614	Erase attempted on a read/only minidisk file.
ERROR	90685	System Error. Unexpected error returned when erasing a minidisk file.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.
ERROR	98700	DATAONLY option is not supported by this file pool. The file pool server is at the z/VM Version 1 Release 1.0 or earlier level.

DMSERP - Extract/Replace



Notes:

[1] The function keyword – including the colon and four-character environment ID, *envir* – is placed in a character variable padded to eight bytes on the right with blanks.

[2] Up to 10 sets of search arguments can be specified.

Call Format

The format for calling a CSL routine is language dependent. DMSERP is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSERP
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSERP" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSERP routine to obtain or update specific subsets of system information.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSERP.

GETENV
(input, CHAR, 8) sets up an Extract/Replace environment that is protected against accidental alteration. (See usage note "1" on page 164.) DMSERP returns a four-character environment ID in the buffer, and an application must present the ID in order to issue requests in that environment. An environment, general or protected, lasts until the next IPL or until it is reset.

RESET
RES:*envir*
(input, CHAR, 8) reinitialize an Extract/Replace environment:

- Flags
- The points at which DMSERP continues searches for particular information names
- Tokens
- Search arguments
- The environment ID, *envir*

RESET affects the general environment, and an application can inadvertently reset the environment for other applications. RES:*envir* affects only the environment identified by *envir*. After the RES:*envir* function is used, *envir* is no longer a valid environment ID.

EXTRACT
EXT:*envir*
(input, CHAR, 8) get system information. EXTRACT functions in the general environment, and the data can be inadvertently altered by other applications' calls to DMSERP, such as RESET. EXT:*envir* applies only to the environment identified by *envir*.

REPLACE
REP:*envir*
(input, CHAR, 8) update system information. REPLACE affects the general environment and can inadvertently change the data returned to other applications. REP:*envir* applies only to the environment identified by *envir*.

numargs
(input, INT, 4) is the number of sets of search arguments being passed on this call. Up to 10 sets of search arguments are allowed on a call to Extract/Replace.

This parameter is required by the GETENV function, but the value is ignored.

infoname
(input, CHAR, 20) is the name of the information to be extracted or replaced. The information name is a character string padded on the right with blanks. If there are no search arguments, DMSERP searches for the next occurrence of this information name in the system.

Supported information names are listed in Appendix C, "Extract/Replace Supported Information Names," on page 577.

This parameter is required by the GETENV function, but the value is ignored.

buffer
(input/output, CHAR, *buflen*) contains the four-character environment ID, or the replacement information, or the extracted information.

datatype
(input/output, INT, 4) is a code for the type of data being extracted or replaced:

Code	Data type
4	Numeric: 4-byte binary
9	Indicator: 1-byte character with a value of '0' or '1'
13	Address: 4-byte unsigned binary

32
Character: fixed-length character string

The GETENV function returns 32. The EXTRACT function returns the code for the data type of the information. The REPLACE function requires the code for the data type of the information being replaced.

buflen
(input/output, INT, 4) is the length in bytes of the information in the buffer.

The actual length of the buffer used by DMSERP is 1024 bytes. DMSERP uses the buffer length parameter to determine whether the data returned can fit in the buffer provided by the application program:

- On a call to to set up a protected environment or to extract, *buflen* must contain the length of the buffer; upon return, it contains the actual length of the extracted information.
- On a call to replace, *buflen* must contain the length of the replacement information: if the data type is Numeric or Address, *buflen* must be at least 4; if the data type is Indicator, *buflen* must be at least 1.

flags
(input, CHAR, 8) is a string, "ABCDEFGH", of flags that control the search process. The flags parameter is optional for information that occurs once, but it is required for information that occurs more than once.

Each flag is either '0' or '1':

Flag
Values

A
'0' = start the search at the beginning of data being searched. '1' = continue search from the point following the last extract/replace match for the information name.

B
'0' = use same search arguments for the continued search. '1' = new search arguments are being passed for the continued search.

If **A** is '0', **B** must also be '0'.

C
'0' = no token is being passed. '1' = token is being passed; do not initiate a search, but look up related information.

If **A** is '1', **C** must be '0'.

DEFGH
are ignored, but they must be either '0' or '1'.

search_type
(input, CHAR, 4) is a keyword that determines how search arguments are combined logically. The keyword must be either AND or OR. If only one set of search arguments is specified, *search_type* is ignored. The keyword must be padded on the right with blanks to a length of 4 bytes.

The search type parameter is optional for information names associated with information that occurs once, but it is required for information names associated with information that occurs more than once.

token
(input/output, INT, 4) is a token for a specific instance of the designated information name.

Extract/Replace uses the token to keep track of where the requested information was found. You can then pass this token on future calls to extract or replace additional information relating to the same information name.

The third flag in the flags parameter (**C**) refers to this token.

The token parameter is optional for information names associated with information that occurs once, but it is required for information names associated with information that occurs more than once.

Extract/Replace allows up to 32767 tokens to be saved in one Extract/Replace environment. When this limit is reached (when return code 124 is received on a DMSERP call), then the environment must be reset with either RESET for the general environment, or RES:*envir* for a protected environment. To continue using a protected environment, a new environment ID must be obtained with GETENV before the DMSERP call is retried.

Search arguments

Search arguments narrow the search through system information. Extract/Replace searches for the next occurrence of a group of related data matching the search arguments. The information names specified for the *infoname* and *search_arg_name* parameters must belong to the same information name set (see Appendix C, “Extract/Replace Supported Information Names,” on page 577 for a set of tables containing these names). If the information associated with the information name specified for *infoname* is not contained in the group of related information matching the search arguments, 133 is returned in *retcode* to indicate that no information was found.

Up to 10 sets of search arguments are allowed per call, each set comprising five parameters. For example, if the *numargs* parameter is 2, there are actually 10 search arguments at the end of the parameter list. The five search argument parameters are:

search_arg_name
(input, CHAR, 20) is a variable for specifying an information name used to qualify the search for the information name in the *infoname* parameter. The information name must be padded on the right with blanks. See Appendix C, “Extract/Replace Supported Information Names,” on page 577 for a complete list of information names.

search_arg_val
(input, *search_val_type*, *search_val_len*) is a value to be compared with the system information designated in *search_arg_name*.

search_val_type
(input, INT, 4) is the data type of the value contained in *search_arg_val*. This data type must match the data type of the information associated with *search_arg_name*.

The codes are the same as for the *datatype* parameter:

Code
Data type

4
Numeric: 4-byte binary

9
Indicator: 1-byte character with a value of ‘0’ or ‘1’

13
Address: 4-byte unsigned binary

32
Character: fixed-length character string

search_val_len
(input, INT, 4) is the length, in bytes, of the value contained in *search_arg_val*. The length must be less than or equal to the length of the information associated with *search_arg_name*.

search_compare_type
(input, CHAR, 2) is a code for the type of comparison to be made between the system data and the search value. Possible values are:

EQ (equal)
GT (greater than)
LT (less than)

GE (greater than or equal)
LE (less than or equal)
NE (not equal)

Usage Notes

1. The EXTRACT, REPLACE, and RESET functions work in the general Extract/Replace environment. The general environment is accessible to all programs, and a program can alter and even reset it while another program is using it. A program should use a protected environment if it needs to leave an Extract/Replace environment undisturbed for itself or another program.

   The first Extract/Replace call in a program should be:

   - RESET, unless the program depends on the general Extract/Replace environment left by another program
   - GETENV, if the program needs to use a protected environment

   See the *z/VM: CMS Application Development Guide* for examples of using DMSERP.
2. The type of information extracted is indicated by the code returned in the data type parameter. The information extracted is returned in the buffer parameter. You must specify the length of the buffer when you extract system information, so that Extract/Replace can determine if the buffer is large enough to contain the extracted information. The value in *buflen* is changed by Extract/Replace to the actual length of the information returned.

   For example:

   - If *datatype* is 32 (Character data),
     - *buflen* contains the length of the data in *buffer*.
     - *buffer* contains the character data.
   - If *datatype* is 4 (Numeric Data),
     - *buflen* is 4.
     - The first 4 bytes of *buffer* contain a fullword numeric value.
   - If *datatype* is 9 (Indicator data),
     - *buflen* is 1.
     - The first byte of *buffer* is character '0' or '1'.
     - '0' indicates that the information had a value or meaning of false, off, or no.
     - '1' indicates that the information had a value or meaning of true, on, or yes.
   - If *datatype* is 13 (Address data),
     - *buflen* is 4.
     - The first 4 bytes of *buffer* contain an address or pointer.
3. When you replace system information, the *datatype* parameter must contain the code for the type of data being replaced, and the *buflen* parameter must contain the length, in bytes, of the replacement data in the buffer.

   For example:

   - If *datatype* is 32 (Character data),
     - *buflen* contains the length of the replacement data in *buffer*.
     - *buffer* contains the replacement character data.
   - If *datatype* is 4 (Numeric Data),
     - *buflen* is 4.
     - *buffer* contains the replacement fullword numeric data.

- If *datatype* is 9 (Indicator data),
  - *buflen* is 1.
  - The first byte of *buffer* is character '0' or '1'.
  - '0' indicates that the information should be set to a value or meaning of false, off, or no.
  - '1' indicates that the information should be set to a value or meaning of true, on, or yes.
- If *datatype* is 13 (Address Data),
  - *buflen* is 4.
  - *buffer* contains the replacement fullword address data.

4. Some information occurs only once in system storage. An example is the size of your virtual storage. In contrast, a search for the access modes of disks or directories matching some search criteria could yield several access modes. The tables in Appendix C, "Extract/Replace Supported Information Names," on page 577 indicate which system information occurs only once.
5. When you call DMSERP with an information name associated with information that occurs only once, you can omit the flags, search type, and token parameters, since their contents are ignored for such information. (Search arguments must be omitted.)
6. The number of parameters you should pass is: 7, for an information name that occurs only once; or 10 plus 5 times the value in *numargs*, for an information name that occurs more than once.
7. The *search_arg_val*, *search_val_len*, and *search_val_type* parameters are closely related. The value contained in the *search_val_type* parameter indicates the type of data contained in the *search_arg_val* parameter. The data type in *search_val_type* must match that of the associated *search_arg_name*.

   For example:

   - If *search_val_type* is 32 (Character data),
     - *search_arg_val* contains the search character data.
     - *search_val_len* contains the length of the search data in *search_arg_val*.
   - If *search_val_type* is 4 (Numeric Data),
     - *search_val_len* is 4.
     - *search_arg_val* contains the search numeric data.
   - If *search_val_type* is 9 (Indicator data),
     - *search_val_len* is 1.
     - The first byte of *search_arg_val* is character '0' or '1':

       '0' indicates that the search is for information that has a value of false, off, or no.
       '1' indicates that the search is for information that has a value of true, on, or yes.
     - *search_compare_type* is either "EQ" or "NE".
8. Information names that return a numerical value, such as ACT_DISK_FILE_SYSTEM and ACT_DISK_CONTROL_LVL, require that *search_compare_type* be EQ or NE.
9. If your application makes a continued search, uses new search arguments, or uses the token function, it must not issue any system commands or in any way alter system information between calls to Extract/Replace.
10. REXX works only with character data, so when DMSERP is called from a REXX program, use the C2D and D2C REXX built-in functions described in the *z/VM: REXX/VM Reference* to convert character format data to numeric format and back. A typical program flow for extracting information would be:

    a. Call DMSERP to get the data you want.
    b. Check the value in parameter 6 (*datatype*) to determine the type of the extracted data.
    c. If the data type is character or indicator, the data can be used directly.

d. If the data type indicates numeric or address data, the data must be converted to REXX character format using C2D(LEFT(*buffer,buflen*)).

If you are replacing numeric or address information, the replacement value contained in *buffer* must be converted from REXX character format to numeric format using D2C(*buffer*,4).

If you are passing in a numeric or address search value, it must be converted from REXX character to numeric format using D2C(*search_arg_val*,4).

11. Extract/Replace does not distinguish between empty files and files that are not empty. As with any files that are opened new, empty files do not have record length data unless they have been opened with a specific record length.
12. Using File Set Information Names, you can extract information about files, aliases, and external objects in accessed SFS directories. However, aliases of erased files, aliases whose authority has been revoked, and external objects do not have any file data. If you try to extract file data information (such as record format, or file block information), you get an error return code indicating that the information is not available.

Return Codes

The normal return codes from DMSERP are:

Code
Meaning

0
Extract/Replace completed the request successfully.

130
You have reached the last occurrence, or there were no occurrences, of data for the information name. You have not been using search arguments.

131
You have reached the last occurrence, or there were no occurrences, of data matching your search arguments.

132
No data currently exists for the set that contains your information name. See Appendix C, “Extract/Replace Supported Information Names,” on page 577 to determine what set your information name belongs to.

133
The information name is not applicable to the instance of related data matching the search arguments. You can continue searching. An example of a situation that would cause this return code is an explicit request for the FILE_DIRECTORY_ID of a minidisk file (an explicit request requires that search arguments be specified).

134
The data returned in the buffer may not reflect the final conditions after asynchronous I/O has been completed.

Error Conditions

In the error return codes from DMSERP, *nn* indicates the position in the parameter list of an incorrect parameter: *retcode* is always parameter number 01, the function is number 02, and so on.

Code
Meaning

100
The information name that was passed to Extract/Replace is invalid.

101
You cannot replace the data for this information name. You can only extract it.

102
The data type does not match the actual data type of the data you are trying to replace. The correct data type has been passed back in *datatype*. See Appendix C, "Extract/Replace Supported Information Names," on page 577 for the correct data type.

103
The length of the buffer in *buflen* is not long enough to contain the data you are trying to extract. The minimum buffer length required to perform this extract has been passed back in *buflen*. See Appendix C, "Extract/Replace Supported Information Names," on page 577 for the correct length. The buffer must be at least this long.

104
The length of the buffer parameter in *buflen* is longer than the actual data you are trying to replace. The maximum replacement data length allowed to perform this replace has been returned in *buflen*. See Appendix C, "Extract/Replace Supported Information Names," on page 577 for the correct length.

105
There is only one occurrence of the information name you are trying to extract or replace. Therefore, you cannot do a continued search, pass new search arguments, or pass a token for this information name. The first three characters of the *flags* parameter must be set to '0'.

106
You have not previously done an extract or replace operation on the information name. Therefore, you cannot do a continued search or pass new search arguments. Make sure that the first two characters of the *flags* parameter are '0's.

Extracting or replacing an information name using a token does not count for the purpose of doing a continued search. Also, you cannot do a continued search based on operations performed before a RESET operation.

107
The length of the replacement data you supplied in the *buflen* parameter is 0. The maximum replacement data length allowed to perform this replace has been passed back in the *buflen* field.

108
You tried to replace an information name with a data type of indicator. The value passed in the first character of *buffer* was not a character 0 or 1.

109
The token flag in the flags parameter is set to '1' but the value in the *token* parameter is invalid. One of the following has happened:

1. You have not yet done an extract or replace operation on the information name or a related information name.
2. You did not save the token passed back from the previous extract or replace operation, or you changed the token to an invalid value.
3. You have done a reset operation since the last extract or replace operation.

110
You have set both the continued-search flag and the token flag (first and third characters in the flags parameter). Only one of these characters can be set to '1' at a time.

111
You have not set the continued-search flag but you have set the flag indicating that you are passing in new search arguments for a continued search (first character in the flags parameter set to '0' and the second character set to '1'). This is an invalid combination.

112
The number of parameters was incorrect. Either 1) you did not pass seven parameters on a GETENV call; or 2) you did not pass exactly two parameters on a RESET call; or 3) the total number of parameters passed did not match the number you should have passed based on the number of sets of search arguments you indicated in *numargs*. See usage note "6" on page 165.

113
You passed more than one set of search arguments but the value in *search_type* was not AND or OR.

117
The function parameter was not GETENV, EXTRACT, EXT:*envir*, REPLACE, REP:*envir*, RESET, or RES:*envir*.

119
One of the characters in the *flags* parameter was not a '0' or '1'.

121
You passed search arguments on a call to extract or replace an information name that occurs only once. In this case, search arguments are not allowed.

123
You tried to pass more than 10 sets of search arguments on a call to extract or replace information as indicated in *numargs*. Ten is the maximum number of sets of search arguments allowed.

124
Extract/Replace allows up to 32767 tokens to be saved. You have exceeded this limit. Before retrying your call, reset the Extract/Replace environment with RESET or RES:*envir* and do not use any tokens returned by Extract/Replace on previous calls.

If you want to continue using a protected environment, you must get a new environment ID with GETENV.

125
The environment ID passed as part of the function keyword does not identify an environment.

500-511
System error. Extract/Replace storage has been overwritten.

10*nn*
The *nn*th parameter in your call was declared with an incorrect data type. Check the explanation of each parameter to determine what the correct data type should be.

20*nn*
The *nn*th parameter in your call was declared with an incorrect length. Check the explanation of each parameter to determine what the correct length should be.

30*nn*
The *nn*th parameter contained an invalid search argument name. See Appendix C, "Extract/Replace Supported Information Names," on page 577 for a complete list of information names.

40*nn*
The *nn*th parameter contained a search argument name which was invalid for the information name being extracted or replaced. Search argument names must come from the same information set as the information name being extracted or replaced. See Appendix C, "Extract/Replace Supported Information Names," on page 577 for valid search arguments for each information set.

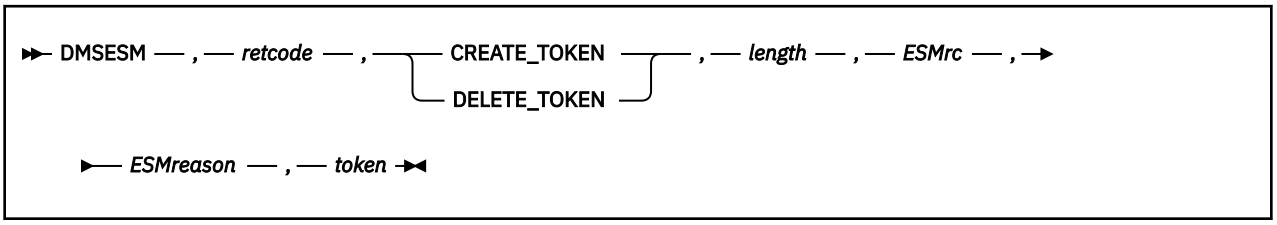
50*nn*
The *nn*th parameter (*search_compare_type*) contained an invalid comparison type. Valid comparison types are EQ, NE, LT, LE, GT, or GE.

60*nn*
The *nn*th parameter (*search_compare_type*) contained an invalid comparison type. Valid comparison types for information names that return indicator or numerical values are EQ or NE.

70*nn*
The *nn*th parameter (*search_arg_name*) contained an information name that cannot be used as a search argument. Appendix C, "Extract/Replace Supported Information Names," on page 577 shows which information names can be used as search arguments.

80*nn*
The *nn*th parameter (*search_arg_val*) contained an invalid search value with a data type of indicator. All search values of data type indicator must consist of a single character, either '0' or '1'.

DMSESM - Identify Program to External Security Manager



Call Format

The format for calling a CSL routine is language dependent. The routine name is the first parameter in DMSCSL's parameter list:

DMSESM
(input, CHAR, 8) can be passed as a literal or in a variable. DMSESM must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSESM routine to identify your program to an External Security Manager (ESM). Your program will receive an identifying token for use with other ESM-related CSL routines or the RACROUTE macro.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSESM.

CREATE_TOKEN
(input, CHAR, 12) establishes a security environment within the ESM and obtains a token by which that environment may be identified on subsequent ESM-related calls.

DELETE_TOKEN
(input, CHAR, 12) releases the specified security token and deletes the ESM environment the token represents.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (CREATE_TOKEN or DELETE_TOKEN).

ESMrc
(output, INT, 4) is a variable for the return code from the external security manager.

ESMreason
(output, INT, 4) is a variable for the reason code from the external security manager.

token
(input/output, CHAR, 4) is a variable for a security token.

For a CREATE_TOKEN request, if *retcode*, *ESMrc*, and *ESMreason* are zero, DMSESM returns a value in *token* that may be used on other ESM-related calls such as DMSLINK and DMSPWCHK.

For a DELETE_TOKEN request, you should specify a value in *token* that was returned by a previous CREATE_TOKEN request.

Usage Notes

1. The ESM environment must be established before a call to DMSESM is made. For RACF/VM, this is done using the RPIUCMS INIT command. Other ESMs may have their own procedures.

2. DMSESM communicates with the external security manager using the RACROUTE REQUEST=VERIFY macro interface. A nonzero ACEE is expected to be returned for all successful CREATE_TOKEN requests. Additional information about this interface can be found in the *z/VM: Security Server RACROUTE Macro Reference*.
3. TCP/IP for z/VM provides an RPIDUMY command that can be used in place of RPIUCMS. This command returns a nonzero token, but will respond "defer" if any subsequent RACROUTE macro calls are made.

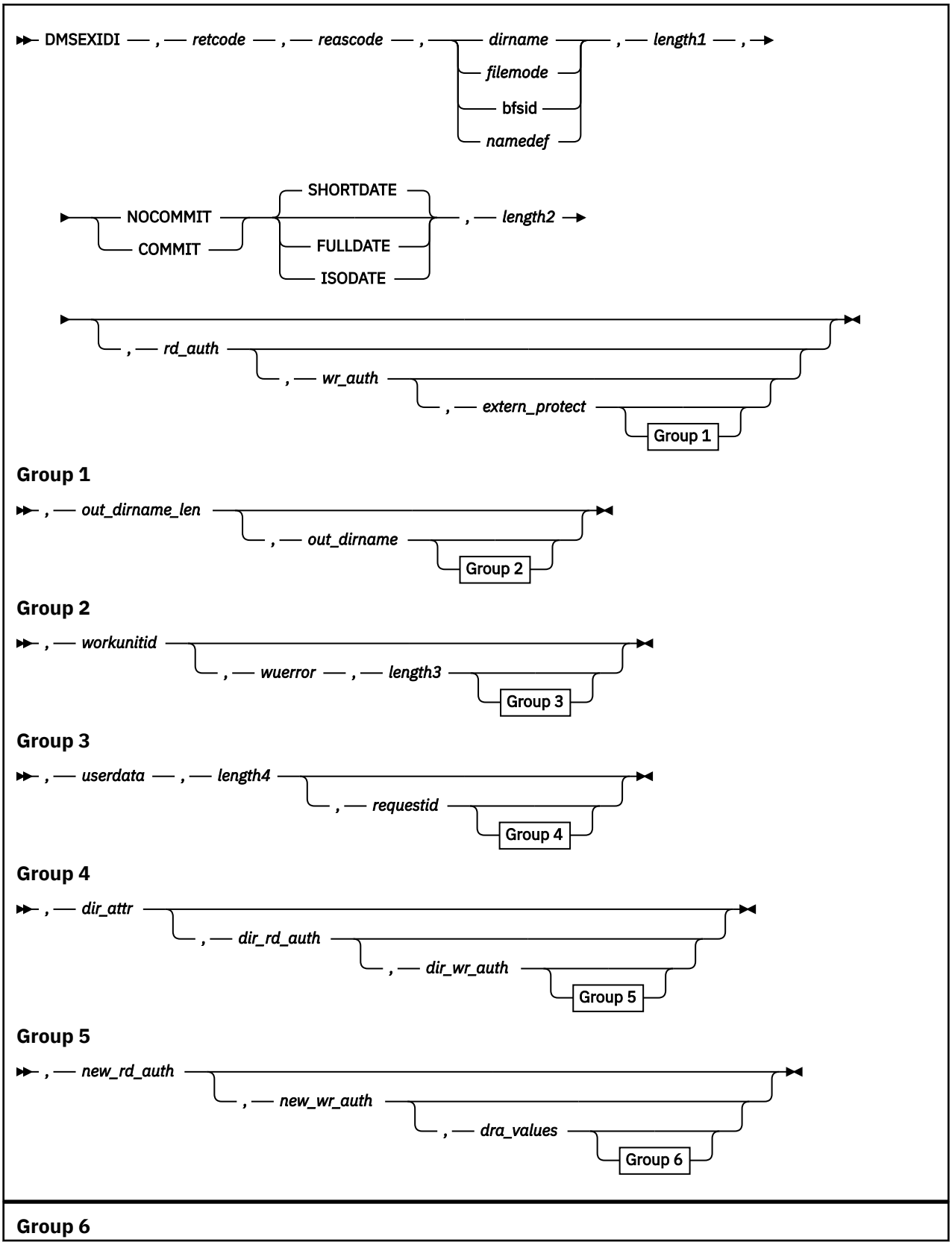
Return Codes and Reason Codes

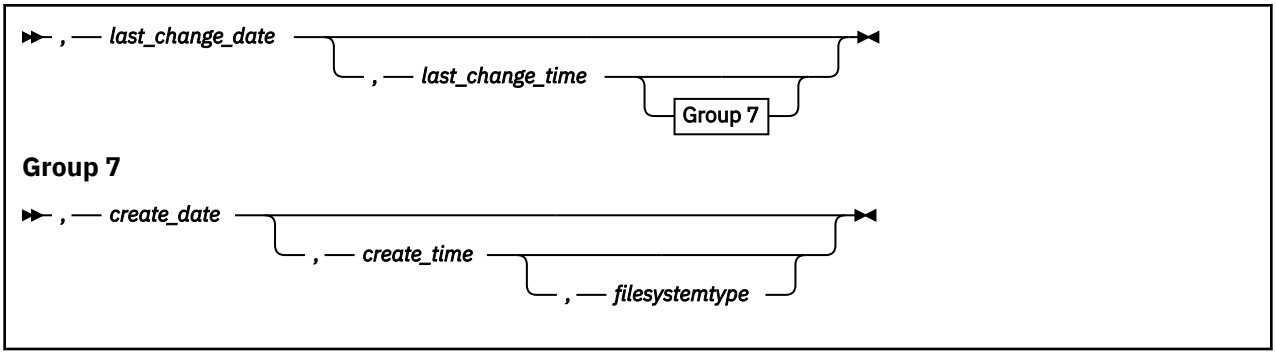
The following table lists the DMSESM return codes.

Return Code	Description
0	Function completed successfully. For CREATE_TOKEN requests, *token* is filled in.
4	Function could not be performed. The ESM is not available or the communications interface is not initialized. See the *ESMrc* for details.
8	Function completed unsuccessfully. See the *ESMrc* and *ESMreason* for details.
-1*nn*	parameter *nn* is not valid

External security manager return and reason codes are specific to the security product being used. For RACF/VM, refer to Usage Note "2" on page 170. For other security products, consult the product documentation.

DMSEXIDI - Exist - Directory





Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSEXIDI is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSEXIDI
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSEXIDI routine to check for the existence of an SFS directory or byte file system (BFS top directory) and to retrieve directory-related information.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSEXIDI.

reascode
(output, INT, 4) is a variable for the reason code from DMSEXIDI.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the SFS directory to be verified.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed SFS directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef*) rather than a file mode. If the file mode is not an accessed SFS directory, an error return code will result.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system (BFS top directory) to be verified.

namedef
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname* or *bfsid*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*dirname*, *filemode*, *bfsid*, or *namedef*).

COMMIT
(input, CHAR, 6) means keep all changes made during a work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

SHORTDATE
(input, CHAR, 9) indicates the format of the *last_change_date* and *create_date* parameters is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *last_change_date* and *create_date* parameters is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *last_change_date* and *create_date* parameters is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE). See "Compound Variables" on page 15 for coding details.

rd_auth
(output, CHAR, 1) is a variable in which a value is returned that indicates whether you have read authority:

0
false

1
true

wr_auth
(output, CHAR, 1) is a variable in which a value is returned that indicates whether you have write authority:

0
false

1
true

extern_protect
(output, CHAR, 1) is a variable in which an indicator is returned to show whether the directory is externally protected:

0
indicates no External Security Manager (ESM) protection.

1
indicates ESM protection.

out_dirname_len
(output, INT, 4) is a variable in which the length of the *out_dirname* parameter is passed back.

out_dirname
(output, CHAR, 153) is a variable in which the directory name is passed back. The actual length of the directory name is passed back in the preceding *out_dirname_len* parameter.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

dir_attr
(output, CHAR, 1) is a variable in which the value of the directory attribute is returned:

0
indicates that the directory is an SFS file control directory or a BFS directory.

1
indicates that the directory is an SFS directory control directory.

dir_rd_auth
(output, CHAR, 1) is a variable in which a value is returned that indicates whether you have directory control read (DIRREAD) authority:

0
indicates that you do not have DIRREAD authority.

1
indicates that you have DIRREAD authority.

dir_wr_auth
(output, CHAR, 1) is a variable in which a value is returned that indicates whether you have directory control write (DIRWRITE) authority when the directory is not accessed. When the directory is accessed, this value indicates whether the directory is accessed in read/write status.

0
indicates that you do not have DIRWRITE authority. Or, if the directory is accessed, it is accessed in read-only status.

1
indicates that you have DIRWRITE authority.

new_rd_auth
(output, CHAR, 1) is a variable in which an indication of your NEWREAD authority is returned:

0
indicates that you do not have NEWREAD authority. This is the value returned for a directory control directory.

1
indicates that you have NEWREAD authority.

new_wr_auth
(output, CHAR, 1) is a variable in which an indication of your NEWWRITE authority is returned:

0
indicates that you do not have NEWWRITE authority. This is the value returned for a directory control directory.

1
indicates that you have NEWWRITE authority.

dra_values
(output, CHAR, 24) is a variable in which the three 8-byte DFSMS/VM related attribute fields are returned. If none are available, zeros are returned.

last_change_date
(output, CHAR, 8 or 10) is a variable in which the Coordinated Universal Time (UTC) date of the last change is returned. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

last_change_time
(output, CHAR, 8) is a variable in which the Coordinated Universal Time (UTC) of the last change is returned. The format is *hh:mm:ss*.

create_date
(output, CHAR, 8 or 10) is a variable in which the Coordinated Universal Time (UTC) date the directory was created is returned. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

create_time
(output, CHAR, 8) is a variable in which the Coordinated Universal Time (UTC) the directory was created is returned. The format is *hh:mm:ss*.

filesystemtype
(output, CHAR, 1) is a variable in which a value indicating the file system type is returned:

0
indicates the object is in an SFS file space.

1
indicates the object is in a BFS file space.

Usage Notes

1. For file control directories, the issuer of DMSEXIDI must have at least read authority on the specified directory.

   For directory control directories, the issuer of DMSEXIDI must have at least directory read (DIRREAD) authority on the specified directory.

   If the caller is not properly authorized, a *not found* condition is returned.

   For BFS directories, the issuer of DMSEXIDI must be a file pool administrator.

2. The value returned in *dir_wr_auth* is influenced by the access status of the directory. If, for example, you have DIRWRITE authority and have the directory accessed in read-only status, *dir_wr_auth* will indicate a '0'. If you release the directory (or access it in read/write status) and execute DMSEXIDI again, *dir_wr_auth* will indicate a '1'.
3. A successful return code indicates that the directory exists, and the requested information is returned in the output parameters.
4. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the specified directory.
5. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Exist - Directory operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
6. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
7. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
8. File pool servers that are not at the current release level do not support all of the DMSEXIDI parameters. Rather than return error or warning codes, DMSEXIDI tolerates down-level servers where possible. For the unsupported parameters, DMSEXIDI returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note "7" on page 194, in the description of DMSEXIST.
9. The time and date of creation and the time and date of the last change are recorded in Coordinated Universal Time (UTC).
10. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to both the *last_change_date* and *create_date* parameters.
11. If the CSL routine is called from a program written in REXX, the *last_change_date* and *create_date* fields returned will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, these fields will be padded on the right with 2 blanks.
12. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSEXIDI, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSEXIDI. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

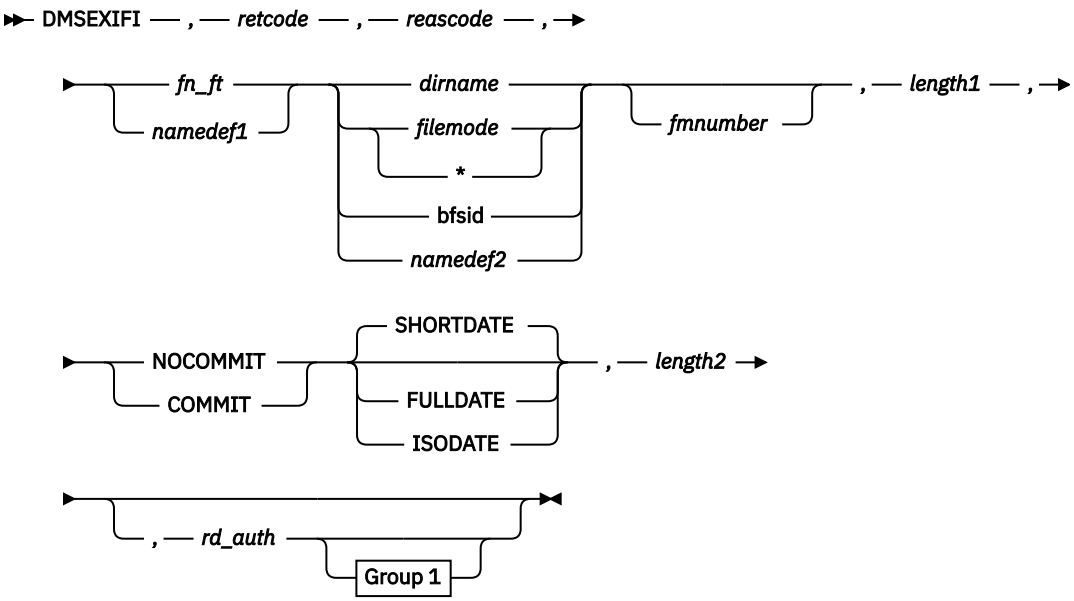
Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

DMSEXIDI can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

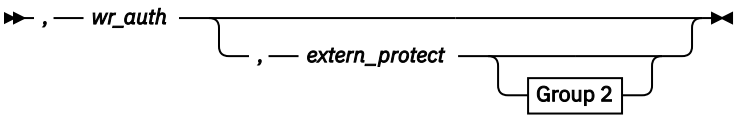
Severity	Reason Code	Description
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces while committing or rolling back a work unit.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	44000	The specified directory does not exist or you are not authorized for it.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90255	A file name, file type, or file mode number may not be specified in a *dirname* parameter.
ERROR	90300	Invalid input parameter in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and no more than four tokens in the string.
ERROR	90410	Invalid length specified for one of the character variables.

Severity	Reason Code	Description
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *dirname* parameter does not exist or was used incorrectly. For example, a temporary name (*namedef*) for directory name was used where a temporary name for a file was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

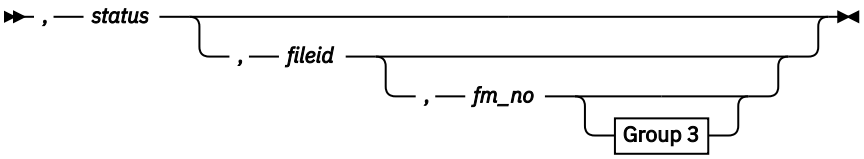
DMSEXIFI - Exist - File



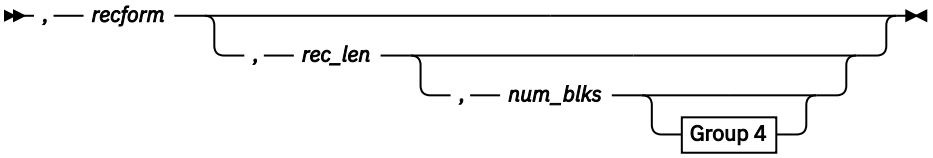
Group 1



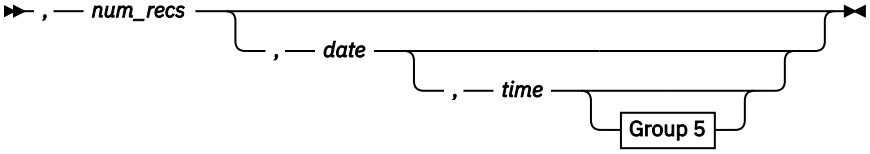
Group 2



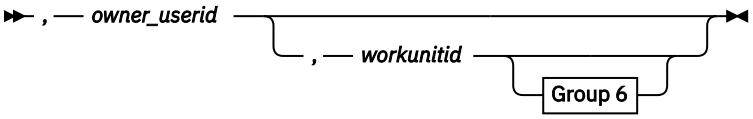
Group 3



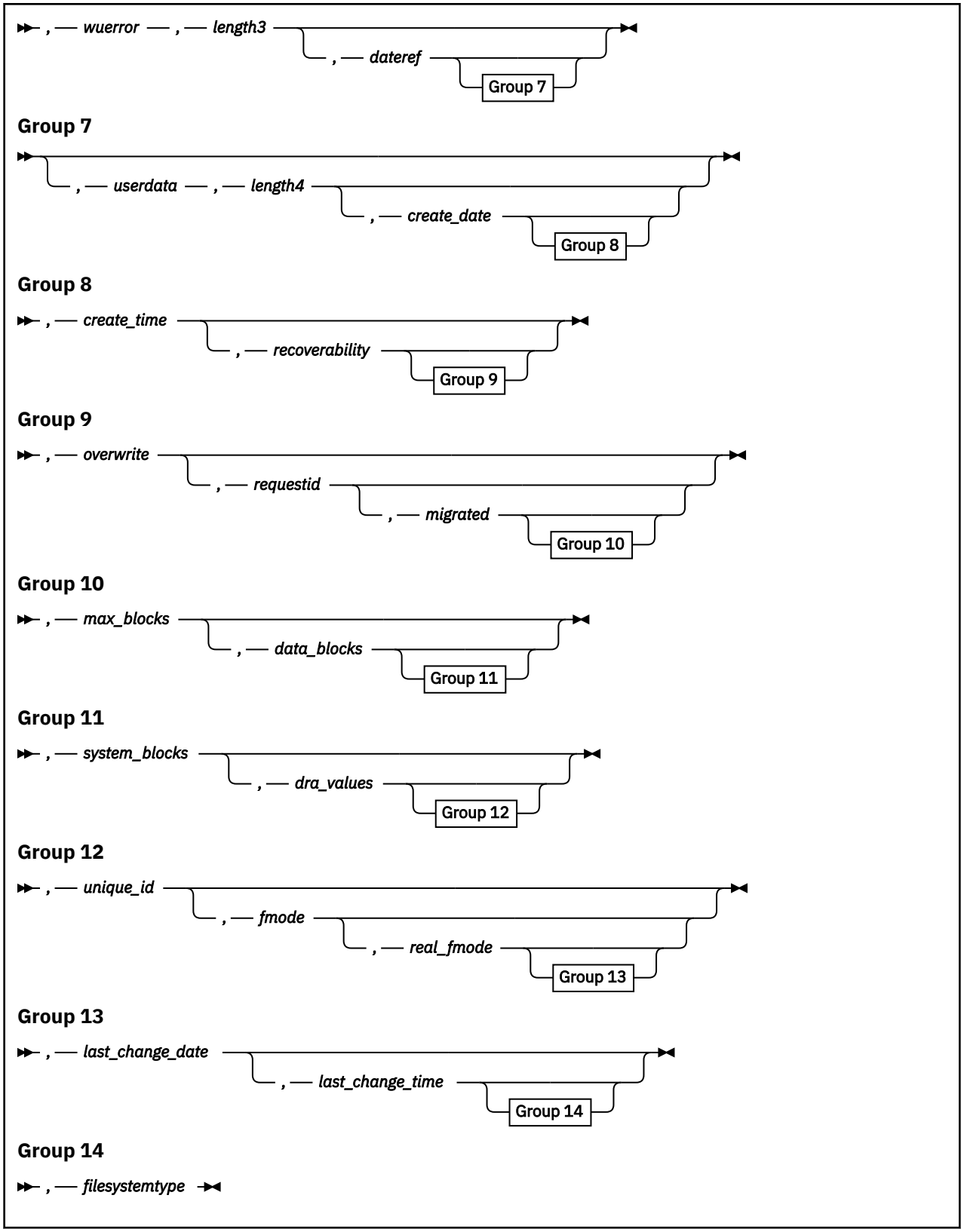
Group 4



Group 5



Group 6



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSEXIFI is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSEXIFI
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSEXIFI routine to check for the existence of a file, alias, or external object and to retrieve file-related information.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSEXIFI.

reascode
(output, INT, 4) is a variable for the reason code from DMSEXIFI.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the name of the file, alias, or external object that is to be checked. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the SFS directory to search for the specified file.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed minidisk or SFS directory to be searched. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode.

An asterisk (*) may be used to specify searching of all accessed minidisks and SFS directories. If an asterisk is specified, the file modes are searched in their normal A-Z search order. If a file mode letter has an accessed file mode extension, this is also searched.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system to be searched for the specified file.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

fmnumber
(input, CHAR, 1) is a variable for specifying the numeric part of the file mode. It is a character variable between 0 and 6. If it is specified but it does not match the file mode number for *fn_ft*, a warning is returned. If not specified and the file is found, the return code is 0.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*; plus *dirname*, *filemode*, *bfsid*, or *namedef2*; plus *fmnumber*, if specified). See "Compound Variables" on page 15 for coding details.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date*, *dateref*, *create_date*, and *last_change_date* parameters is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date*, *dateref*, *create_date*, and *last_change_date* parameters is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date*, *dateref*, *create_date*, and *last_change_date* parameters is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

rd_auth
(output, CHAR, 1) is a variable in which a value is returned that indicates whether you have read authority:

0
false

1
true

wr_auth
(output, CHAR, 1) is a variable in which a value is returned that indicates whether you have write authority:

0
false

1
true. If minidisk, then accessed as read/write.

extern_protect
(output, CHAR, 1) is a variable in which a value is returned that indicates whether the file is protected by an External Security Manager (ESM):

0
indicates no ESM protection.

1
indicates there is ESM protection.

For CMS minidisk files, and DOS or OS formatted minidisk files, this parameter always has a value of '0'.

status
(output, CHAR, 1) is a variable in which a value is returned that indicates the type of the file:

1
SFS base file or BFS regular file

2
alias

3
erased alias

4
revoked alias

6
external object

7
minidisk file

8
file on OS or DOS formatted minidisk

fileid
(output, CHAR, 16) is a variable in which the file name and file type are returned as two adjacent 8-byte fields.

fm_no
(output, CHAR, 1) is a variable in which the file mode number is returned.

recform
(output, CHAR, 1) is a variable in which a value is returned that indicates whether the file contains fixed-length or variable-length records or has been erased. Possible values are:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

- (hyphen)
indicates that the file is an erased or revoked alias or is an external object (see the *status* parameter to determine which it is).

' ' (blank)
indicates that the file is either an OS or DOS formatted minidisk file.

rec_len
(output, INT, 4) is a variable in which the record length for the file is returned. (0 is returned for external objects and OS or DOS formatted minidisk files.)

num_blks
(output, INT, 4) is a variable in which the number of blocks contained in the file is returned. (0 is returned for external objects and OS or DOS formatted minidisk files.)

num_recs
(output, INT, 4) is a variable in which the number of records contained in the file is returned. (0 is returned for external objects and OS or DOS formatted minidisk files.)

date
(output, CHAR, 8 or 10) is a variable in which is returned the local date the file was last updated or the external object was created. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. It is blanks if the file is a DOS or OS formatted minidisk file.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could have storage overlays.

time
(output, CHAR, 8) is a variable in which is returned the time the file was last updated or the external object was created, in the character format *hh:mm:ss*. It is blanks if the file is a DOS or OS formatted minidisk file.

owner_userid
(output, CHAR, 8) is a variable in which the user ID of the owner of the file or external object is returned. It is blanks for a minidisk file.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to use an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable for an area in which extended error information is returned from CMS. If this parameter is used, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

dateref
(output, CHAR, 8 or 10) is a variable in which is returned the Universal Coordinated Time (UTC) date of the last reference to the file or external object (either created, modified, or read). It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. Blanks are returned for external objects and minidisk files.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could have storage overlays.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see usage note "3" on page 186).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

create_date
(output, CHAR, 8 or 10) is a variable in which the Coordinated Universal Time (UTC) date on which the file was created is returned. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. This will be blanks if this is a minidisk file.

For the 4-digit years, either the FULLDATE or ISODATE parameters must be specified. SHORTDATE is the default, which is the 2-digit year format.

create_time
(output, CHAR, 8) is a variable in which the Universal Coordinated Time (UTC) time at which the file was created is returned. The time is returned in the character format *hh:mm:ss*. This will be blanks if this is a minidisk file.

recoverability
(output, CHAR, 1) is a variable in which a value is returned that indicates whether a file is to be reset to its last committed state during rollback processing:

1
RECOVER

0
NORECOVER

- (hyphen)
Not applicable because the file has been erased or revoked or is an external object. See the *status* parameter to determine which it is.

' ' (blank)
This indicates that the file is either an OS or DOS formatted minidisk file.

overwrite
(output, CHAR, 1) is a variable in which a value is returned that indicates whether a reader of the file will see a consistent version of the file throughout the duration of processing:

0
NOTINPLACE

1
INPLACE

- (hyphen)
Not applicable because the file has been erased or revoked or is an external object. See the *status* parameter to determine which it is.

' ' (blank)
This indicates that the file is either an OS or DOS formatted minidisk file.

requestid
(input/output, INT, 4) is a variable containing a value that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

migrated
(output, CHAR, 1) is a variable in which a value is returned that indicates whether the file has been placed in migrated status by DFSMS/VM:

0
indicates the file is not in DFSMS/VM migrated status.

1
indicates the file is in DFSMS/VM migrated status.

' ' (blank)
indicates there are objects with a *status* value other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

max_blocks
(output, INT, 4) is a variable in which the largest block number for the file is returned. (A zero is returned for external objects and minidisk files.)

data_blocks
(output, INT, 4) is a variable in which the number of blocks used for the file (data only) is returned.. (A zero is returned for external objects and minidisk files.)

system_blocks
(output, INT, 4) is a variable in which is returned the number of additional blocks the system needs for the file in addition to the data blocks. (A zero is returned for minidisk files.)

dra_values
(output, CHAR, 24) is a variable in which three 8-byte DFSMS/VM attribute fields are returned. If none are available, the fields are returned as zeros. Blanks are returned for external objects and minidisk files. An alias has the DFSMS/VM-related attributes of its base file. The values in the DFSMS/VM-related attribute fields of erased and revoked aliases are zeros.

unique_id
(output, CHAR, 16) is a variable in which a value is returned that uniquely identifies an object within a file pool: base file, alias, alias of an erased file, revoked alias, or external object. It contains blanks for a minidisk file. See usage note “4” on page 186.

Certain administrative actions, such as reorganizing the file pool catalogs, can cause unique ID values to change, but the value for each object is always unique within the file pool.

fmode
(output, CHAR, 1) is a variable in which is returned the file mode letter found for the file. This parameter contains the file mode letter of the input minidisk or SFS file when the input specifies a file mode letter or an asterisk. If *dirname* is specified on the input, this parameter contains the file mode letter of the input directory, if it is accessed. If it is not accessed, this is blank.

real_fmode
(output, CHAR, 1) is a variable in which is returned the file mode letter that the file is actually contained on. This is different from *fmode* when the real file mode is an extension of the input file mode. If *dirname* is specified on the input, this parameter contains the file mode letter of the input directory, if it is accessed. If it is not accessed, this is blank.

last_change_date
(output, CHAR, 8 or 10) is a variable in which the Universal Coordinated Time (UTC) date of the last change to the file, external object, or alias is returned. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. Blanks are returned for minidisk files.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

last_change_time
(output, CHAR, 8) is a variable in which the Coordinated Universal Time (UTC) of the last change to the file, external object, or alias is returned. The format is *hh:mm:ss*. Eight blanks are returned for minidisk files.

filesystemtype
(output, CHAR, 1) is a variable in which a value is returned that indicates the file system type:

0
indicates that the object is in an SFS file space.

1
indicates that the object is in a BFS file space.

2
indicates that the file is on a minidisk.

Usage Notes

1. To issue DMSEXIFI on a minidisk file, the user must have the minidisk accessed.

   To issue DMSEXIFI on an SFS file, the user must have at least read authority on *fn_ft* or on the parent directory. If the user is not properly authorized, a *not found* condition is returned.

   To issue DMSEXIFI on a BFS file, the user must be a file pool administrator. DMSEXIFI is valid only for BFS regular files.
2. A successful return code indicates that the file exists, and the requested information is returned in the parameters.
3. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the specified file. If the directory being opened is minidisk, *userdata* is not used.
4. Besides DMSEXIFI, CSL routines DMSEXIST, DMSGETDF, DMSGETDI, and DMSGETDX return the unique ID of an SFS file. You can use DMSEXIST to obtain information about file pool objects identified by the unique ID.

5. File pool servers that are not at the current release level do not support all of the DMSEXIFI parameters. Rather than return error or warning codes, DMSEXIFI tolerates down-level servers where possible. For the unsupported parameters, DMSEXIFI returns values that are consistent with the functions provided by the down-level server. These parameters are listed in usage note "7" on page 194 in the description of the DMSEXIST routine.
6. If issued on a revoked or erased alias, DMSEXIFI returns zeros or blanks for the file attributes.
7. The date of last reference, creation date and time, and date and time of last change (*dateref*, *create_date*, *create_time*, *last_change_date*, and *last_change_time* fields) are recorded in Coordinated Universal Time (UTC). The date and time of last update (*date* and *time* fields), however, are local time for the CMS user machine.
8. For an alias, the date of last reference (*dateref* field) is not updated when either the alias or its base file is referenced. A reference to an alias updates the date of last reference of its base file without altering the date of last reference of the alias. At the time of its creation, an alias acquires the date of last reference of its base file, and this value does not change. For information about using the date of last reference attribute in your program, see the *z/VM: CMS Application Development Guide*.
9. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Exist - File operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
10. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
11. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
12. All minidisk requests are done synchronously.
13. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date*, *dateref*, *create_date*, and *last_change_date* parameters.
14. If DMSEXIFI is called from a program written in REXX, the *date*, *dateref*, *create_date*, and *last_change_date* fields returned will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, these fields will be padded on the right with 2 blanks.
15. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.
16. A DMSEXIFI request on a BFS object results in the following output parameter values:

Table 19. DMSEXIFI Output Values for a BFS Object

Parameter	Output Value
status	1, indicating a BFS regular file
fm_no	1, indicating a file mode number of 1
recform	F, indicating fixed-length records
rec_len	1, indicating a record length of 1 byte
num_recs	Number of bytes in the file **Note:** If the file contains more than 2^{31}-1 bytes, a value of -1 is returned.
recoverability	1, indicating RECOVER
overwrite	0, indicating NOTINPLACE

Table 19. DMSEXIFI Output Values for a BFS Object (continued)	
Parameter	**Output Value**
system_blocks	Number of additional blocks generated by the system
owner_userid	BFS file space name

Return Codes and Reason Codes

For lists of the possible return codes from DMSEXIFI, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSEXIFI. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

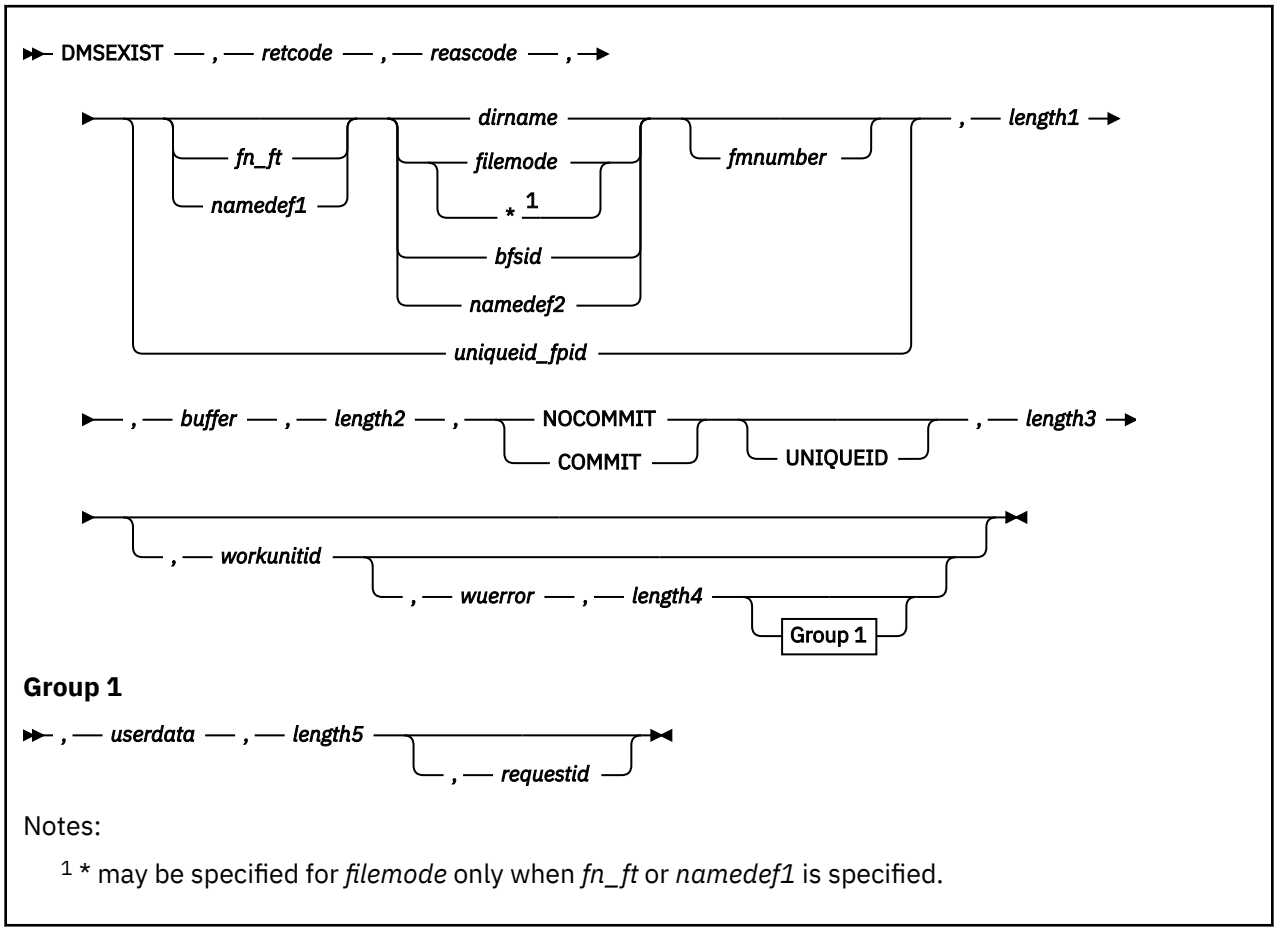
DMSEXIFI can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
WARNING	90200	Specified file mode number does not match the file mode number of the specified file or external object.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	44000	The specified file, directory, or external object does not exist or you are not authorized for it.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.

Severity	Reason Code	Description
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90250	The file name and file type (or *namedef*) are required but were not specified.
ERROR	90300	Invalid input parameter in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.
ERROR	90410	Invalid length specified for one of the character variables.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.
ERROR	90440	The specified file mode number is invalid. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a temporary name (*namedef*) for a directory name was used where a temporary name for a file was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90680	I/O error accessing OS dataset.

Severity	Reason Code	Description
ERROR	90681	OS read password protected dataset.
ERROR	90682	OS dataset organization is not BSAM, QSAM or BPAM.
ERROR	90683	OS dataset more than 16 extents.
ERROR	90685	Received an unexpected return code during a search for a minidisk file.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSEXIST - Exist



Notes:

[1] * may be specified for *filemode* only when *fn_ft* or *namedef1* is specified.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSEXIST is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSEXIST
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSEXIST routine to check for the existence of one of the following objects:

- SFS file (base file, alias, revoked alias, or alias for an erased file)
- BFS regular file
- Minidisk file
- SFS directory
- Byte file system (BFS top directory)
- External object

Information about the file, directory, or external object is returned in a buffer.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSEXIST.

reascode
(output, INT, 4) is a variable for the reason code from DMSEXIST.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file or external object that is to be checked. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the SFS directory name. If *fn_ft* or *namedef1* is specified, this is the directory containing the file or external object that is to be checked. If neither *fn_ft* nor *namedef1* is specified, this is the directory that is to be checked.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed minidisk or SFS directory to be searched. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode.

An asterisk (*) may be used to specify searching of all accessed minidisks and SFS directories. (If neither *fn_ft* nor *namedef1* is specified, an asterisk is not allowed.) If an asterisk was specified, the file modes are searched in the usual A-Z search order. If the file mode letter has an accessed file mode extension, this is also searched.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system. If *fn_ft* or *namedef1* is specified, this identifies the byte file system containing the file that is to be checked. If neither *fn_ft* nor *namedef1* is specified, this identifies the byte file system (BFS top directory) that is to be checked.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number. It is a character value from '0' to '6', inclusive. If *fmnumber* is specified but does not match the file mode number for the file or external object, a warning is returned. If *fmnumber* is not specified and the file or external object is found, the return code is 0. The *fmnumber* field is ignored for directories.

uniqueid_fpid
(input, CHAR, 17-24) is a variable for identifying a file pool object by its unique ID and file pool. The object can be an SFS base file, a BFS regular file, an alias, an alias of an erased file, a revoked alias, an external object, or a directory, but it must be committed. An uncommitted object that is identified by its unique ID cannot be found. If the object is found, a file or directory record is returned.

The content of *uniqueid_fpid* is a 16-character unique ID and a 1- to 8-character file pool ID. The IDs are concatenated with no delimiting blank, and there is no colon at the end of the file pool ID. When the object to be checked is identified by a unique ID, you must also specify the UNIQUEID keyword to indicate the content of the third parameter.

length1
(input, INT, 4) is a variable for specifying the length of the third parameter, which identifies the object to be checked. The parameter contains either *uniqueid_fpid* or a compound character parameter composed of *fn_ft* or *namedef1*; plus *dirname*, *filemode*, *bfsid*, or *namedef2*; plus *fmnumber*, if

specified. If the character parameter contains a unique ID and file pool ID, *length1* must be no less than 17 and no more than 24. See "Compound Variables" on page 15 for coding details.

buffer
(output, CHAR, *length2*) is an area where the information on the file, external object, or directory is to be returned. See usage note "19" on page 197 for a description of the record formats and the length of each.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

UNIQUEID
(input, CHAR, 8) is a keyword indicating that the object to be checked is identified by a unique ID (see the *uniqueid_fpid* parameter). It is specified in a compound character parameter with COMMIT or NOCOMMIT.

length3
(input, INT, 4) is a variable for specifying the length of thepreceding compound character parameter (COMMIT or NOCOMMIT, and UNIQUEID). See "Compound Variables" on page 15 for coding details.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length5* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. Rather than provide a structure for the information returned in the buffer, you can use the Exist - Directory (DMSEXIDI) and Exist - File (DMSEXIFI) routines to get the information you need instead of using DMSEXIST.

The assembler mapping macro EXSBUFF is also available to map the output.

2. These CSL routines return the unique ID of an SFS file or directory: DMSEXIST, DMSEXIFI, DMSGETDI, DMSGETDF, and DMSGETDX. Certain administrative actions, such as reorganizing the file pool catalogs, can cause unique ID values to change, but the value for each object is always unique within the file pool.
3. To issue DMSEXIST on a minidisk file, the user must have the minidisk accessed.

   To issue DMSEXIST on an SFS file, the user must have at least read authority on *fn_ft* or on the parent directory. To issue DMSEXIST on an SFS directory, the user must have at least read authority (READ or DIRREAD) on the directory. If the user is not properly authorized, a *not found* condition is returned.

   To issue DMSEXIST on a BFS file, the user must be a file pool administrator. DMSEXIST is valid only for BFS regular files.
4. If your program was coded prior to this release, it may not have defined a buffer large enough to hold a complete record. DMSEXIST returns fields in groups related to the releases in which they were added to the record, from oldest to newest. This includes any reserved fields used to pad the end of the record for that release. Therefore DMSEXIST returns the fields for the latest release whose record can fit in the specified buffer without truncation. In any case, DMSEXIST gives you the length of the returned record in the length field. To avoid causing a compatibility problem with your program, DMSEXIST does not return a warning reason code for this case.
5. A successful return code indicates that the file, directory, or external object exists, and the requested information is returned in the *buffer* parameter.
6. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the specified file, directory, or external object.

   If the directory being opened is minidisk, *userdata* is not used.
7. File pool servers that are not at the current release level may not support these file attributes:

Table 20. Later File Attributes

Attribute	**Field Names**
recoverability	recoverability
overwrite	overwrite
UTC date of last reference	dec_dateref, dateref, dec_dateref_ext, dateref_ext, iso_dateref_ext
UTC creation date	dec_cr_date, cr_date, dec_cr_date_ext, cr_date_ext, iso_cr_date_ext
creation time	dec_cr_time, cr_time
maximum blocks	max_blocks
data blocks	data_blocks
system blocks	system_blocks
migration indicator	migrated
DFSMS/VM-related attributes	dra_values
unique ID	unique_id
directory control read authority	dir_rd_auth
directory control write authority	dir_wr_auth

Table 20. Later File Attributes (continued)	
Attribute	**Field Names**
directory attribute	dir_attr
new read authority	new_rd_auth
new write authority	new_wr_auth
UTC date of last change	dec_last_change_date, last_change_date, dec_last_change_date_ext, last_change_date_ext, iso_last_change_date_ext
UTC time of last change	dec_last_change_time, last_change_time
file system type	file_system_type
local date of last update	date, date_ext, dec_date, dec_date_ext, iso_date_ext

If the file pool server does not support an attribute, zeros (integer or character) or blanks are returned in the FILE data record for that attribute. For example, if date of last reference is not supported, the *dateref* field contains the character string "00/00/00", and the *dateref_ext* field contains the character string "0000/00/00". If the migration indicator is not supported, a character zero is returned in the *migrated* field. If the unique ID is not supported, blanks are returned in the *unique_id* field.

Zeros are also returned if the file does not yet have the attribute established for it (perhaps the server has only recently added support for the attribute).

8. When it is issued on a revoked or erased alias, DMSEXIST returns zeros or blanks for the file attributes in the FILE data record.
9. When *uniqueid_fpid* refers to an alias, DMSEXIST returns the FILE record for the base file.
10. The system sets the date of last reference, creation date and time, and date and time of last change to Coordinated Universal Time (UTC). This affects the following fields:
    - *dateref*
    - *dateref_ext*
    - *dec_dateref*
    - *dec_dateref_ext*
    - *iso_dateref_ext*
    - *cr_date*
    - *cr_date_ext*
    - *dec_cr_date*
    - *dec_cr_date_ext*
    - *iso_cr_date_ext*
    - *cr_time*
    - *dec_cr_time*
    - *last_change_date*
    - *last_change_date_ext*
    - *dec_last_change_date*
    - *dec_last_change_date_ext*
    - *iso_last_change_date_ext*
    - *last_change_time*

- *dec_last_change_time*

  The local date and time of last modification are the date and time at the location of the CMS user machine. This affects the following fields:

  a. *date*
  b. *date_ext*
  c. *time*
  d. *dec_date*
  e. *dec_date_ext*
  f. *iso_date_ext*
  g. *dec_time*

11. For an alias, the date of last reference (*dateref*, *dateref_ext*, *dec_dateref*, *dec_dateref_ext*, and *iso_dateref_ext* fields) is not updated when either the alias or its base file is referenced. A reference to an alias updates the date of last reference of its base file without altering the date of last reference of the alias. At the time of its creation, an alias acquires the date of last reference of its base file, and this value does not change. For information about using the date of last reference attribute in your program, see the *z/VM: CMS Application Development Guide*.
12. If the directory is accessed, the *dir_rd_auth* and *dir_wr_auth* values in the directory data record indicate whether the directory is accessed in read-only or read/write status. A '1' for *dir_rd_auth* indicates read-only access, while '1' for *dir_wr_auth* indicates read/write access. When the directory is not accessed, the values indicate what authorities you have on the directory.
13. If the COMMIT parameter is specified and the return code is 8, then either:
    - An error occurred during the processing of the Exist operation, or
    - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
14. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
15. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
16. All minidisk requests are done synchronously.
17. If the file pool server is running at a level prior to VM/ESA® Version 2 Release 1.1, 4-digit years are not supported. In that case, for the dates whose field type is CHAR (10) or INTEGER (4), the 4-digit year returned is always 19*yy*. This affects the following fields:
    - *dec_date_ext*
    - *date_ext*
    - *iso_date_ext*
    - *dec_dateref_ext*
    - *dateref_ext*
    - *iso_dateref_ext*
    - *dec_cr_date_ext*
    - *cr_date_ext*
    - *iso_cr_date_ext*
    - *dec_last_change_date_ext*
    - *last_change_date_ext*
    - *iso_last_change_date_ext*

18. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.
19. If the *fn_ft* parameter is specified with a directory name or file mode, information is returned in a FILE data record, which can refer to a file, an alias, or an external object. If the directory name or file mode is specified without the *fn_ft* parameter, information is returned in a DIRECTORY data record. If the *uniqueid_fpid* parameter is specified, either a FILE or DIRECTORY record is returned.

    The data records are described in Table 21 on page 197 and Table 22 on page 200.

Table 21. Format for FILE Data Record. FILE Record Length: 436 bytes

Offset	Field Name	Field Type	Description
0 (X'0')	type	char(1)	1=FILE
1 (X'1')	file_system_type	char(1)	0= SFS file space, 1= BFS file space, 2= minidisk file
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	fileid	char(16)	file identifier can be a file name (char(8)) and file type (char(8))
20 (X'14')	fm_no	char(1)	{0...6}
21 (X'15')	recform	char(1)	F=fixed, V=variable, -=erased, or external object, (blank)=OS or DOS formatted minidisk file
22 (X'16')	recoverability	char(1)	1=RECOVER, 0=NORECOVER, -=N/A, (blank)=OS or DOS formatted minidisk file
23 (X'17')	overwrite	char(1)	1=INPLACE, 0=NOTINPLACE, -=N/A, (blank)=OS or DOS formatted minidisk file
24 (X'18')	rec_len	integer	length of records in the file (0 for external objects and OS or DOS formatted minidisk files)
28 (X'1C')	num_blks	integer	number of blocks in the file (0 for external objects and OS or DOS formatted minidisk files)
32 (X'20')	num_recs	integer	number of records in the file (0 for external objects and OS or DOS formatted minidisk files)
36 (X'24')	dec_date	integer(3)	Local date of last update in decimal format ("*yymmdd*"). For external objects, this is date of creation. For OS or DOS formatted minidisk files, this is zero.
39 (X'27')	file_mode	char(1)	The file mode letter found for the file or directory if a file mode letter or asterisk was specified. If a directory name was specified, the field contains the file mode letter of the directory if it is accessed, or a blank if the directory is not accessed. If the file was identified by a unique ID, this field contains a blank.

Table 21. Format for FILE Data Record. FILE Record Length: 436 bytes *(continued)*			
Offset	**Field Name**	**Field Type**	**Description**
40 (X'28')	dec_time	integer(3)	Local time of last update in decimal format (*hhmmss*). For external objects, this is time of creation. For OS or DOS formatted minidisk files, this is zero.
43 (X'2B')	real_file_mode	char(1)	The file mode letter at which the file is actually accessed. This is different from the *file_mode* (offset 39) when the actual file mode is an extension of the input file mode. If a directory name was specified, this field contains the file mode letter of the directory, if it is accessed. If it is not accessed, this field is blank. If the file was identified by a unique ID, this field contains a blank.
44 (X'2C')	date	char(8)	Local date of last update in character format "*yymmdd*". For external objects, this is date of creation. For OS or DOS formatted minidisk files, this is blank.
52 (X'34')	time	char(8)	Local time of last update in character format "*hh:mm:ss*". For external objects, this is time of creation. For OS or DOS formatted minidisk files, this is blank.
60 (X'3C')	owner_userid	char(8)	ID of base file or directory owner. Blanks indicate a minidisk file.
68 (X'44')	status	char(1)	1= SFS base or BFS regular, 2= alias, 3= erased, 4= revoked, 6= external object, 7= minidisk file, 8= OS or DOS disk
69 (X'45')	rd_auth	char(1)	read authority: 0= false, 1= true, 0= external object
70 (X'46')	wr_auth	char(1)	write authority: 0= false, 1= true, 0= external object. If minidisk, 1 indicates accessed read/write.
71 (X'47')	extern_protect	char(1)	0= none (or external object or minidisk file), 1= protected
72 (X'48')	dec_dateref	integer(3)	UTC date of last reference in decimal format ("*yymmdd*") (0 for external objects and minidisk files).
75 (X'4B')		integer(1)	reserved
76 (X'4C')	dateref	char(8)	UTC date of last reference in character format “*yy/mm/dd*”. (blank for external objects and minidisk files.)
84 (X'54')	dec_cr_date	integer(3)	UTC creation date in decimal format ("*yymmdd*"). 0 = minidisk file.
87 (X'57')		integer(1)	reserved

Table 21. Format for FILE Data Record. FILE Record Length: 436 bytes *(continued)*

Offset	Field Name	Field Type	Description
88 (X'58')	dec_cr_time	integer(3)	creation time in decimal format (*hhmmss*). 0 = minidisk file.
91 (X'5B')		integer(1)	reserved
92 (X'5C')	cr_date	char(8)	UTC creation date in character format (“*yy/mm/dd*”) (blank) = minidisk file.
100 (X'64')	cr_time	char(8)	UTC creation time in character format (“*hh:mm:ss*”) (blank) = minidisk file.
108 (X'6C')	max_blocks	integer(4)	largest block number for the file. (0 for external objects and minidisk files.)
112 (X'70')	data_blocks	integer(4)	number of blocks used for the file data *only*. (0 for external objects and minidisk files.)
116 (X'74')	system_blocks	integer(4)	number of blocks used by the system for the file in addition to the data blocks. (0 for minidisk files.)
120 (X'78')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM-migrated status; (blank) = external object and minidisk files, objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.
121 (X'79')	dra_values	char(24)	contains the three 8-byte fields for DFSMS/VM-related attributes. Blank for external objects and minidisk files.
145 (X'91')	unique_id	char(16)	value that is unique for each object in a file pool: base file, alias, alias of an erased file, revoked alias, or external object. Blank = minidisk file
161 (X'A1')	dirname_len	integer(1)	length of the directory name (dirname). 0 = minidisk file.
162 (X'A2')	dirname	char(153)	directory name in the format *filepoolid:userid.n1...*; (blank) = minidisk file
315 (X'13B')	dec_last_change_date	integer(3)	UTC date of last change in decimal format ("*yymmdd*"). 0 = minidisk file.
318 (X'13E')		integer(1)	reserved
319 (X'13F')	dec_last_change_time	integer(3)	UTC time of last change in decimal format ("*hhmmss*"). 0 = minidisk file.
322 (X'142')		integer(1)	reserved
323 (X'143')	last_change_date	char(8)	UTC date of last change in character format (“*yy/mm/dd*”). Blank = minidisk file.

Table 21. Format for FILE Data Record. FILE Record Length: 436 bytes *(continued)*			
Offset	**Field Name**	**Field Type**	**Description**
331 (X'14B')	last_change_time	char(8)	UTC time of last change in character format (“*hh:mm:ss*”). Blank = minidisk file.
339 (X'153')	dec_date_ext	integer(4)	local date of last update in decimal ("*yymmdd*"). For external objects, this is date of creation. For OS or DOS formatted minidisk files, this is zero.
343 (X'157')	date_ext	char(10)	local date of last update in character format ("*yymmdd*"). For external objects, this is date of creation. For OS or DOS formatted minidisk files, this is blank.
353 (X'161')	iso_date_ext	char(10)	local date of last update in character format ("*yyyy-mm-dd*"). For external objects, this is date of creation. For OS or DOS formatted minidisk files, this is blank.
363 (X'16B')	dec_dateref_ext	integer(4)	UTC date of last reference in decimal format ("*yymmdd*"). Zero ("0") for external objects and minidisk files.
367 (X'16F')	dateref_ext	char(10)	UTC date of last reference in character format ("*yyyy/mm/dd*"). Blank for external objects and minidisk files.
377 (X'179')	iso_dateref_ext	char(10)	UTC date of last reference in character format (). Blank for external objects and minidisk files.
387 (X'183')	dec_cr_date_ext	integer(4)	UTC creation date in decimal format ("*yyyymmdd*"). 0 = minidisk file.
391 (X'187')	cr_date_ext	char(10)	UTC creation date in character format ("*yyyy/mm/dd*"). Blank = minidisk file.
401 (X'191')	iso_cr_date_ext	char(10)	UTC creation date in character format ("*yyyy-mm-dd*"). Blank = minidisk file.
411 (X'19B')	dec_last_change_date _ext	integer(4)	UTC date of last change in decimal format ("*yymmdd*"). 0 = minidisk file.
415 (X'19F')	last_change_date_ext	char(10)	UTC date of last change in character format ("*yyyy/mm/dd*"). Blank = minidisk file.
425 (X'1A9')	iso_last_change_date _ext	char(10)	UTC date of last change in character format ("*yyyy-mm-dd*"). Blank = minidisk file.
435 (X'1B3')		char(1)	reserved

Table 22. Format for DIRECTORY Data Record. DIRECTORY Record Length: 308 bytes			
Offset	**Field Name**	**Field Type**	**Description**
0 (X'00')	type	char(1)	2= DIR
1 (X'1')	file_system_type	char(1)	0= SFS file space, 1= BFS file space

Table 22. Format for DIRECTORY Data Record. DIRECTORY Record Length: 308 bytes *(continued)*			
Offset	**Field Name**	**Field Type**	**Description**
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	dirname_len	integer(1)	length of the directory name (dirname)
5 (X'05')	dirname	char(153)	directory name in the format: *filepoolid:userid.n1...*
158 (X'9E')	rd_auth	char(1)	read authority: 0= false, 1= true
159 (X'9F')	wr_auth	char(1)	write authority: 0= false, 1= true
160 (X'A0')	extern_protect	char(1)	0= none, 1= protected
161 (X'A1')	dir_rd_auth	char(1)	directory control read authority: 0= false, 1= true
162 (X'A2')	dir_wr_auth	char(1)	directory control write authority: 0= false, 1= true
163 (X'A3')	dir_attr	char(1)	directory attribute: 0= SFS FILECONTROL directory or BFS top directory, 1= SFS DIRCONTROL directory
164 (X'A4')	new_rd_auth	char(1)	NEWREAD authority: 0= false, 1= true
165 (X'A5')	new_wr_auth	char(1)	NEWWRITE authority: 0= false, 1= true
166 (X'A6')	dra_values	char(24)	contains three 8-byte fields for DFSMS/VM-related attributes. Blank for external objects.
190 (X'BE')	unique_id	char(16)	value that is unique for each directory in the file pool
206 (X'CE')	dec_last_change_date	integer(3)	UTC date of last change in decimal format ("*yymmdd*").
209 (X'D1')		integer(1)	reserved
210 (X'D2')	dec_last_change_time	integer(3)	UTC time of last change in decimal format ("*hhmmss*").
213 (X'D5')		integer(1)	reserved
214 (X'D6')	last_change_date	char(8)	UTC date of last change in character format ("*yy/mm/dd*")
222 (X'DE')	last_change_time	char(8)	UTC time of last change in character format ("*hh:mm:ss*")
230 (X'E6')	dec_cr_date	integer(3)	UTC creation date in decimal format ("*yymmdd*").
233 (X'E9')		integer(1)	reserved
234 (X'EA')	dec_cr_time	integer(3)	UTC creation time in decimal format ("*hhmmss*").

Table 22. Format for DIRECTORY Data Record. DIRECTORY Record Length: 308 bytes *(continued)*			
Offset	**Field Name**	**Field Type**	**Description**
237 (X'ED')		integer(1)	reserved
238 (X'EE')	cr_date	char(8)	UTC creation date in character format ("*yy/mm/dd*")
246 (X'F6')	cr_time	char(8)	UTC creation time in character format ("*hh:mm:ss*")
254 (X'FE')	dec_last_change_date _ext	integer(4)	UTC date of last change in decimal format ("*yyyymmdd*")
258 (X'102')	last_change_date_ext	char(10)	UTC date of last change in character format ("*yyyy/mm/dd*")
268 (X'10C')	iso_last_change_date _ext	char(10)	UTC date of last change in character format ("*yyyy-mm-dd*")
278 (X'116')	dec_cr_date_ext	integer(4)	UTC creation date in decimal format ("*yyyymmdd*")
282 (X'11A')	cr_date_ext	char(10)	UTC creation date in character format ("*yyyy/mm/dd*")
292 (X'124')	iso_cr_date_ext	char(10)	UTC creation date in character format ("*yyyy-mm-dd*")
302 (X'12E')		char(6)	reserved

20. Output values for BFS objects are as follows:

Table 23. DMSEXIST Output Values for a BFS Object	
Field	**Output Value**
fm_no	1, indicating a file mode number of 1
recform	F, indicating fixed-length records
recoverability	1, indicating RECOVER
overwrite	0, indicating NOTINPLACE
rec_len	1, indicating a record length of 1 byte
num_recs	Number of bytes in the file **Note:** If the file contains more than 2^{31}-1 bytes, a value of -1 is returned.
status	1, indicating a BFS regular file
system_blocks	Number of additional blocks generated by the system
dir_attr	0, indicating a BFS top directory
owner_userid	Name of the BFS file space

Return Codes and Reason Codes

For lists of the possible return codes from DMSEXIST, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSEXIST. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

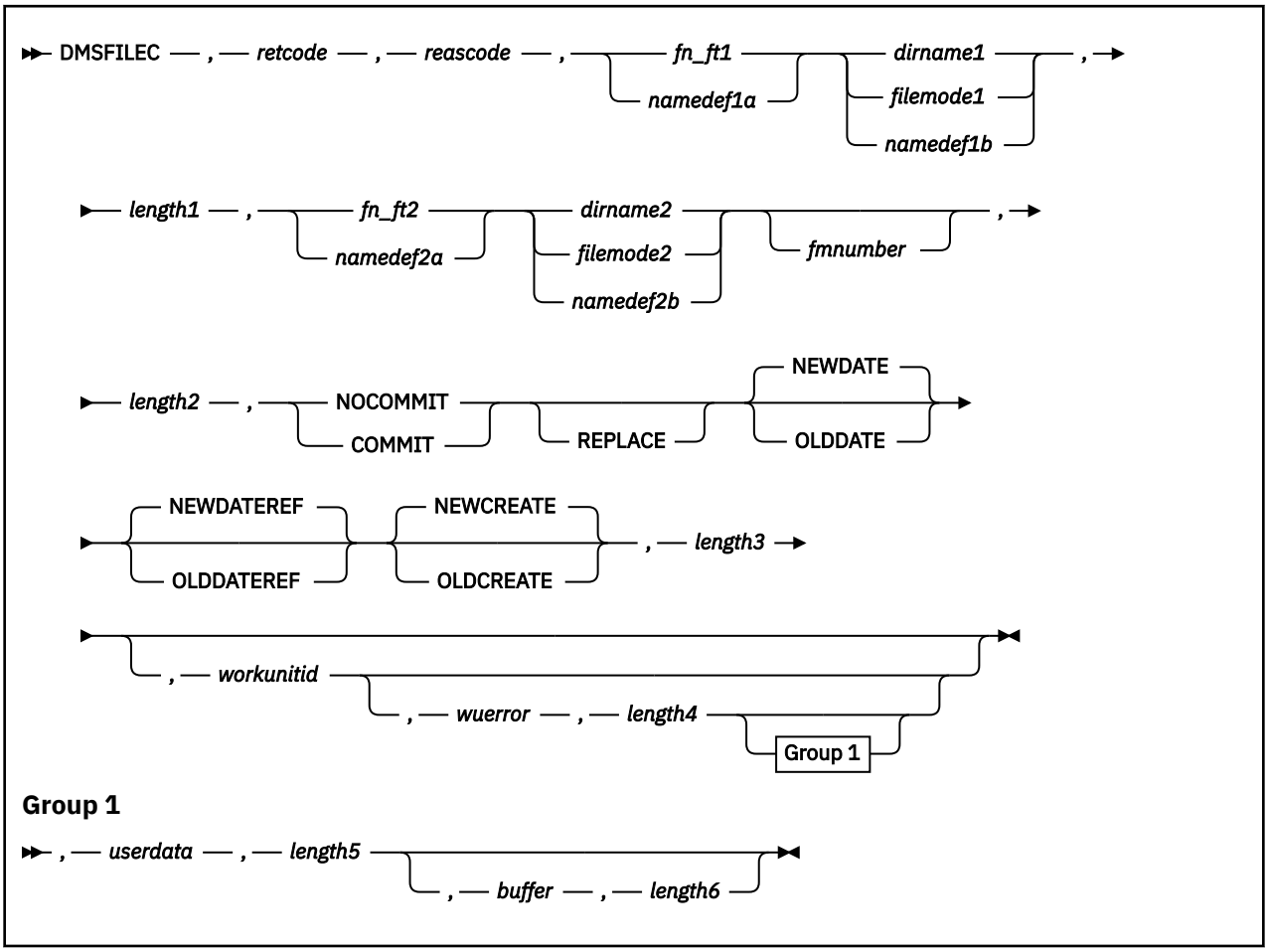
DMSEXIST can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
WARNING	90200	Specified file mode number does not match the file mode number of the specified file or external object.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE. COMMIT was specified.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	44000	The file, alias, external object, or directory specified by the *uniqueid_fpid* parameter has not been committed to the file pool, or you are not authorized to use it.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit; therefore a new logical block is not available.
ERROR	90220	The specified file, external object, or directory does not exist or you are not authorized to it.
ERROR	90230	The specified directory does not exist or you are not authorized to it.
ERROR	90270	Output buffer is too small to contain the requested fixed length output.
ERROR	90300	Invalid parameter specified. The COMMIT/NOCOMMIT parameter must contain "COMMIT" or "NOCOMMIT", possibly followed by "UNIQUEID".

Severity	Reason Code	Description
ERROR	90350	Incorrect number of blank-delimited tokens in the third parameter, which identifies the object to be checked. There must be at least one and not more than four tokens in the string.
ERROR	90410	Incorrect length specified for one of the character variables.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90445	Incorrect characters (* or %) were found in directory name part of the *dirname* parameter.
ERROR	90450	Incorrect characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90455	Incorrect character (* or %) found in file mode or directory fields.
ERROR	90472	Incorrect request ID specified. It must be 0 or 1.
ERROR	90476	Incorrect file pool ID specified in the *uniqueid_fpid* parameter.
ERROR	90481	Incorrect length specified for the *uniqueid_fpid* parameter. The length must be greater than or equal to 17 and less than or equal to 24.
ERROR	90486	An incorrect unique ID—all zeros—specified in the *uniqueid_fpid* parameter.
ERROR	90500	The specified directory name is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Exist for directory function cannot be performed on a minidisk.
ERROR	90680	I/O error accessing OS dataset.
ERROR	90681	OS read password protected dataset.
ERROR	90682	OS dataset organization is not BSAM, QSAM or BPAM.
ERROR	90683	OS dataset more than 16 extents.

Severity	Reason Code	Description
ERROR	90685	Received an unexpected return code during a search for a minidisk file.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSFILEC - Filecopy



Context

File System Management: SFS and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSFILEC is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSFILEC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSFILEC routine to copy one file to another file:

- From minidisk file to minidisk file
- From minidisk file to SFS file
- From SFS file to minidisk file
- From SFS file to SFS file (within a file pool or between file pools)

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSFILEC.

reascode
(output, INT, 4) is a variable for the reason code from DMSFILEC.

fn_ft1
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the source file being copied.

namedef1a
(input, CHAR, 1-16) is a variable for specifying a namedef (temporary name) of the source file being copied.

dirname1
(input, CHAR, 1-153) is a variable for specifying the name of the directory containing the source file being copied.

filemode1
(input, CHAR, 1) is a variable for specifying the file mode of the minidisk or SFS file, to be copied. If this file mode letter is also a one character namedef (*namedef1b*), the namedef will be used.

namedef1b
(input, CHAR, 1-16) is a variable for specifying a namedef (temporary name) of the directory or file mode containing the source file being copied.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft1* or *namedef1a*; plus *dirname1*, filemode1 or *namedef1b*). See "Compound Variables" on page 15 for coding details.

fn_ft2
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the target file with the copied information.

namedef2a
(input, CHAR, 1-16) is a variable for specifying a namedef (temporary name) for the target file with the copied information.

dirname2
(input, CHAR, 1-153) is a variable for specifying the directory name containing the target file with the copied information.

filemode2
(input, CHAR, 1) is a variable for specifying the file mode of the directory or minidisk containing the target file with the copied information. If this file mode letter is also a one character namedef (*namedef1b*), the namedef will be used.

namedef2b
(input, CHAR, 1-16) is a variable for specifying a namedef (temporary name) for the target file with the copied information.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number. This is a character value between 0 and 6. If omitted, the file containing the copied information has the same file mode number as the file being copied.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft2* or *namedef2a*; plus *dirname2*, *filemode2* or *namedef2b*; plus *fmnumber*). See "Compound Variables" on page 15 for coding details.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

REPLACE
(input, CHAR, 7) means to replace the existing file when the file designated to contain the copied information already exists. When REPLACE is specified, the attributes of the source file (for example, record format and logical record length) and the authorities of the target file are kept. If REPLACE is not specified and the target file already exists, the existing file is not replaced and an error code is returned.

OLDDATE
(input, CHAR, 7) specifies that the date and time of last modification for the target file should be derived from the source file.

NEWDATE
(input, CHAR, 7) specifies that the date and time of last modification for the target file should be set to the current date and time. NEWDATE is the default.

OLDDATEREF
(input, CHAR, 10) specifies that SFS should not update the date of last reference for the file being copied (that is, the source file). SFS always updates the date of last reference for the target file.

This parameter is not applicable for a minidisk file.

NEWDATEREF
(input, CHAR, 10) specifies that SFS should update the date of last reference for the file being copied (that is, the source file). SFS always updates the date of last reference for the target file. NEWDATEREF is the default.

This parameter is not applicable for a minidisk file.

OLDCREATE
(input, CHAR, 9) specifies that the creation date and time for a new target file should be those of the source file. (This parameter is ignored for existing target files.)

This parameter is not applicable for a minidisk file.

NEWCREATE
(input, CHAR, 9) specifies that the system should determine the creation date and time for a new target file. (This parameter is ignored for existing target files.) NEWCREATE is the default.

This parameter is not applicable for a minidisk file.

length3
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (COMMIT or NOCOMMIT; plus REPLACE (if specified); plus OLDDATE or NEWDATE; plus OLDDATEREF or NEWDATEREF; plus OLDCREATE or NEWCREATE). See "Compound Variables" on page 15 for coding details.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended error information from CMS. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify

a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length5* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

buffer
(input, CHAR, length6) is the data buffer that will be used by Filecopy when transferring a file. This buffer is useful for applications that call Filecopy multiple times, and do not want to the system incur the overhead of obtaining storage on every call.

length6
(input, INT, 4) is a variable for specifying the length of the *buffer* parameter. This length must be a multiple of 4096 (4KB). Specifying 0 has the effect of omitting the *buffer* parameter.

Usage Notes

1. If DMSFILEC is called with the REPLACE parameter specified to change the file mode number of a base file, then the file mode number of all aliases on that base file are changed. If the target is an alias, then the file mode number of the base file and all other aliases on the base (including the one specified on the call) are changed.
2. If the specified output file does not exist prior to the invocation of DMSFILEC, then the recoverability and overwrite attributes of the output file are determined by the default settings (established by the DMSPUSHA (Push Attributes) routine) corresponding to the file mode number of the output file. If no default attributes setting has been established, the file will be created with the RECOVER and NOTINPLACE attributes.
3. If the REPLACE option is specified and the target file exists, the recoverability and overwrite attributes of the file are not modified by this operation.

   A minidisk file always has the RECOVER attribute. The file will be INPLACE if the file mode number is 6.
4. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Filecopy operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
5. If the source file resides in a directory control directory, and you have the directory accessed read-only, the data copied will include only those changes made to the file before you accessed the directory. If you do not have the directory accessed, or you release the directory with the CMS RELEASE command, then all changes committed before you copy the file are included.
6. The copy operation will fail if:
   - Another user has locked the target file or the directory in which it resides (a following usage note shows the cases in which a copy will fail because of locking)
   - The target file resides in a directory control directory, and another user has the directory accessed read/write
   - The target file resides in a directory control directory, and you have the directory accessed read-only.

7. These tables show when a user can or cannot copy a file with DMSFILEC when there is a lock on the file or the directory. *Yes* means the user can copy a file under the specified lock and *No* means he cannot.

User	Lock Mode of File or Directory Copied From		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Issuer	Yes	Yes	Yes
Another User	Yes	Yes	No

User	Lock Mode of File or Directory Copied Into		
	Explicit Share	**Explicit Update**	**Explicit Exclusive**
Issuer	No	Yes	Yes
Another User	No	No	No

   Create Lock creates an explicit SHARE, UPDATE, or EXCLUSIVE lock.
8. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify passwords for the source and target files.

   If the directory being opened is minidisk, *userdata* is not used.
9. If you specify OLDDATEREF, NEWDATEREF, OLDCREATE, or NEWCREATE and execute the routine against a file pool that does not support the related attributes, the parameters are not applicable. These parameters are also not applicable when the target is on a minidisk.
10. If the target SFS file already exists, the creation date and time attributes for that file cannot be changed. For new target files, use the OLDCREATE and NEWCREATE parameters to control whether the file assumes the creation date and time of the source file or whether SFS should determine those attributes.
11. If the source SFS file is an empty file, an empty target file is created.
12. If the SFS file is in DFSMS/VM migrated status, it is recalled unless SET RECALL has been set to OFF.
13. When *wuerror* has been specified on DMSFILEC and an SFS error occurs during the DFSMS/VM file recall process, FPERROR is updated to include the specific reason code of the failing request and the file pool ID of the failing resource.
14. For a minidisk file, when the file is copied all updates to the file have been made. If COMMIT was specified, the work unit will be committed. If an error occurs attempting to commit the work unit the operations associated with that work unit will remain uncommitted but the minidisk file will be copied.
15. The *buffer* parameter should be used for applications that are calling DMSFILEC repetitively for a performance improvement. This buffer must be at least 4096 (4KB) in length. An error will result if the length is not a multiple of 4KB. If the file is being copied within the same file pool, this buffer will offer no performance improvement.
16. The OLDDATEREF | NEWDATEREF parameter will only be used when both the source and target files are on an SFS directory. A warning will result if either the source or target file is on a minidisk.
17. Empty SFS files may be copied to another SFS file, but not to a minidisk. An error will result if the source file is empty and the target file is on a minidisk.
18. DMSFILEC uses a temporary work file called DMSFILEC CMSUT1. If your copy operation ends abnormally, the work file may remain on your minidisk. It will disappear the next time you call DMSFILEC.

Return Codes and Reason Codes

For lists of the possible return codes from DMSFILEC, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSFILEC. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

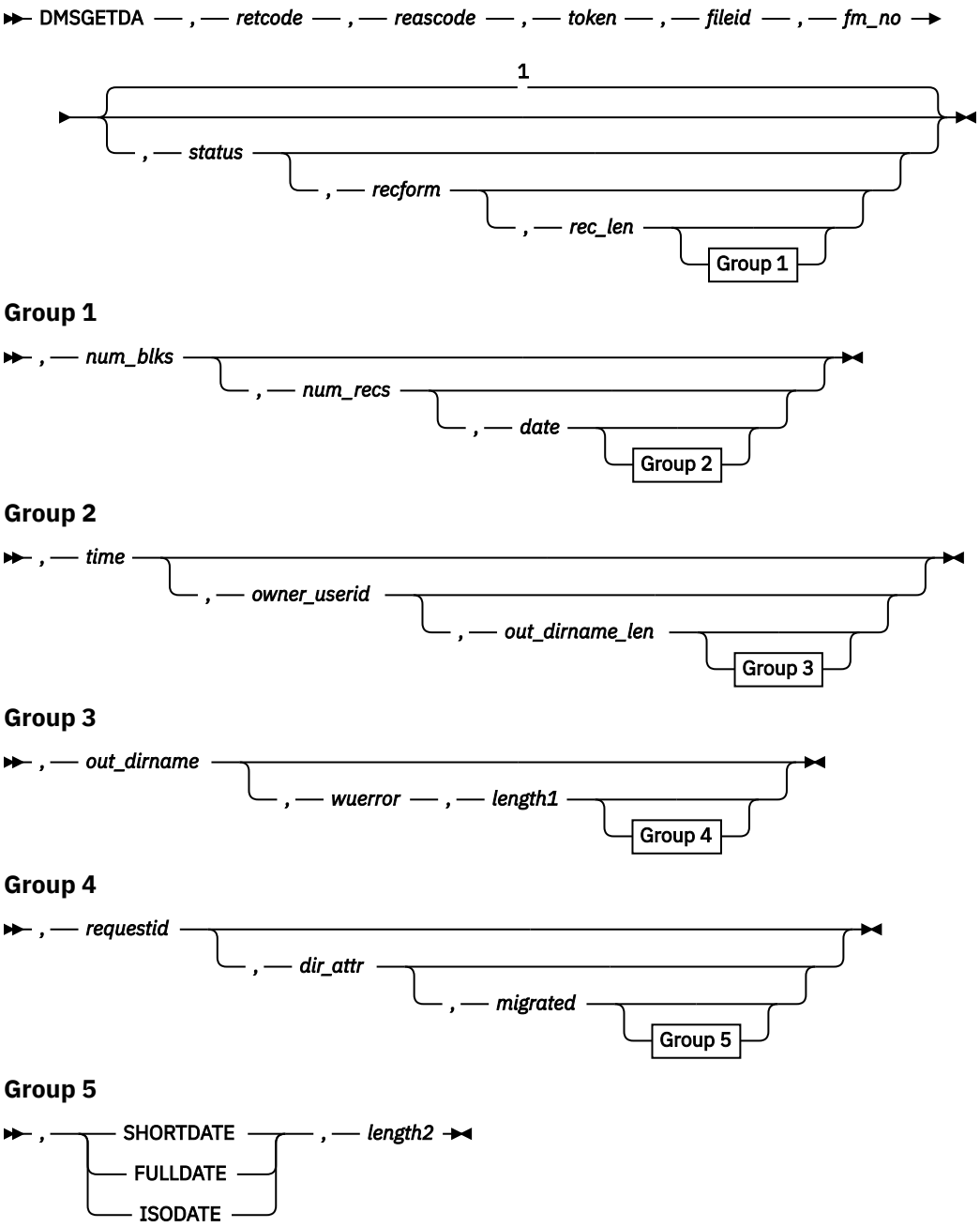
DMSFILEC can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44150	REPLACE was specified, but the target file did not previously exist. The target file was created.
WARNING	51050	File space warning threshold reached or exceeded.
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
WARNING	90623	OLDDATEREF | NEWDATEREF was specified for a minidisk file.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	The target file already exists and REPLACE was not specified.
ERROR	31000	Mixing recoverable and nonrecoverable work for the same file is incorrect within a single work unit.
ERROR	44060	The source file does not exist or you are not authorized to read it.
ERROR	44100	One of the following is true: • Target directory does not exist • Target file does not exist, and you are not authorized to create a file in the target directory • Target file exists, but you are not authorized to write to it
ERROR	44200	The file is already open for WRITE, NEW, or REPLACE using DMSOPEN or DMSOPDBK.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	50600	File cannot be copied. The user has no storage blocks allocated to his file space.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.

Severity	Reason Code	Description
ERROR	63700	The specified DIRCONTROL directory is accessed read-only.
ERROR	64300	The specified target file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.
ERROR	65200	File is in DFSMS/VM migrated status and implicit RECALL is set to OFF. (Set RECALL ON or issue a DFSMS RECALL command.)
ERROR	65400	Specified operation cannot be performed on an external object or on a BFS file.
ERROR	65500	File is in DFSMS/VM migrated and there is no active DFSMS. (The file pool server is running with the NODFSMS startup parameter in effect.)
ERROR	66100	File is in DFSMS/VM migrated status and DFSMS/VM recall processing has been disabled.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT option is specified.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90131	Insufficient minidisk space available.
ERROR	90146	Nonzero bytes follow end of data.
ERROR	90160	Insufficient buffer size. Must be a multiple of 4096 (4K).
ERROR	90250	The file name and file type (or namedef) are required but were not specified.
ERROR	90310	Incorrect option in CSL parameter list. Valid options are COMMIT, NOCOMMIT, REPLACE, OLDDATE, NEWDATE, OLDDATEREF, NEWDATEREF, OLDCREATE, and NEWCREATE.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT, or OLDDATE and NEWDATE, or OLDDATEREF and NEWDATEREF, or OLDCREATE and NEWCREATE were both specified.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and no more than four tokens in the string.
ERROR	90380	Missing parameter in CSL parameter list.
ERROR	90410	Incorrect length specified for one of the character parameters.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Severity	Reason Code	Description
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Special characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90460	File pool IDs in source and target directory are not the same.
ERROR	90500	The specified dirname is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90603	Attempted to open the file for output, but the file mode is read-only.
ERROR	90604	Minidisk file has already been opened with the FS macro interface.
ERROR	90610	Source file was empty and the target file is a minidisk.
ERROR	90684	Attempted to open a file on an OS- or DOS-formatted minidisk.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSGETDA - Get Directory - Searchall



Notes:

[1] Some optional parameters have nonsyntactical default values; see the parameter descriptions.

Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDA is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDA
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDA routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of SEARCHALL. Immediately after Open Directory, Get Directory starts with the directory that was opened. Subsequent calls get the next directory entry until all entries have been retrieved. Then, if there are subdirectories to that directory, their entries are retrieved, and so on down the directory structure.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDA.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDA.

token
(input, CHAR, 8) is a variable for passing the information returned by Open Directory (DMSOPDIR) that uniquely identifies the open directory.

fileid
(output, CHAR, 16) is a variable used to return the file name and file type as two adjacent 8-byte fields, or a subdirectory name.

fm_no
(output, CHAR, 1) is a variable ('0'-'6' or a blank) in which the file mode number is returned.

status
(output, CHAR, 1) is a variable in which the type of the file, directory, or external object is returned:

1
base file

2
alias

3
erased alias

4
revoked alias

5
directory

6
external object

recform
(output, CHAR, 1) is a variable used to return an indicator to show whether the file contains fixed-length or variable-length records, or a directory, or whether the file has been erased. Possible values are:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

D
indicates that this is a directory.

- (hyphen)
indicates that the file is either an erased or revoked alias, or an external object. (see the *status* parameter to determine which it is).

rec_len
(output, INT, 4) is a variable in which the record length for the file is returned. A nonzero value is returned only for base files and aliases.

num_blks
(output, INT, 4) is a variable in which the number of blocks in the file is returned. A nonzero value is returned only for base files and aliases.

num_recs
(output, INT, 4) is a variable in which the number of records in the file is returned. A nonzero value is returned only for base files and aliases.

date
(output, CHAR, 8 or 10) is a variable used to return the date-of-last-update of the file. For external objects, this is the date it was created. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

time
(output, CHAR, 8) is a variable used to return the time the file was last updated or the external object was created, in the character format *hh:mm:ss*.

owner_userid
(output, CHAR, 8) is a variable in which the user ID of the owner of the base file, directory, or external object is returned.

out_dirname_len
(output, INT, 4) is a variable used to return length of the data in the following *out_dirname* parameter.

out_dirname
(output, CHAR, 153) is a variable in which the directory name is passed back. The actual length of the directory name is returned in the preceding *out_dirname_len* parameter.

wuerror
(output, CHAR, *length1*) is a variable used to specify an area for returning the work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If, on return, the request ID is still 1, no server call was needed, and it is not necessary to call DMSCHECK because the function has already been completed.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates a file control directory.

1
indicates a directory control directory.

' ' (blank)
indicates that the returned record does not describe a directory.

migrated
(output, CHAR, 1) is a variable in which a file's migrated status is returned:

0
indicates that file is not in DFSMS/VM migrated status

1
indicates that file is in DFMS/VM migrated status

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. If the date format parameter is not specified, SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (SHORTDATE, FULLDATE, or ISODATE).

Usage Notes

1. This routine returns information about the directory specified on the preceding call to the SFS Open Directory (DMSOPDIR) routine and, on subsequent calls, about each entry in that directory, and then each entry in each subdirectory to which you have authority, except those with exclusive locks. Continued calling of this routine returns all entries within the opened directory structure. When information is returned for the directory that was opened by DMSOPDIR (the first call to DMSGETDA), the *fileid* parameter is left blank.
2. If successful, the requested directory information is returned in the appropriate parameters.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. File pool servers that are not at the current release level do not support all of the DMSGETDA parameters. Rather than return error or warning codes, DMSGETDA tolerates down-level servers where possible. For the unsupported parameters, DMSGETDA returns values that are consistent with

the functions provided by the down-level server. These values are listed in usage note "6" on page 230, in the description of DMSGETDI.

6. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
7. If DMSGETDA is called from a program written in REXX, the *date* field returned will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, this field will be padded on the right with 2 blanks.
8. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

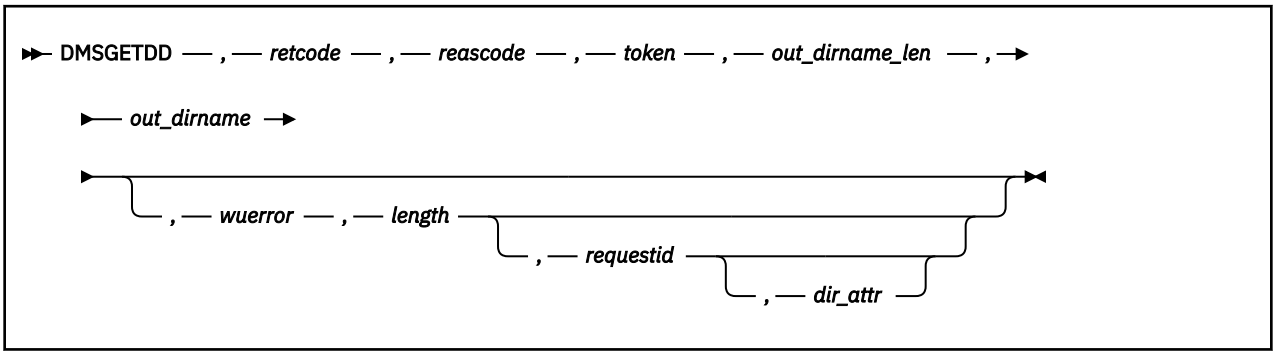
For lists of the possible return codes from DMSGETDA, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSGETDA. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDA can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of SEARCHALL.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90310	Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

DMSGETDD - Get Directory - Dir



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDD
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDD routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of DIR. The first call to DMSGETDD reads the record for the directory that was opened. If it is an SFS directory, each subsequent call to DMSGETDD gets the next subdirectory entry within the open directory. If the open directory is a BFS top directory, information about BFS subdirectories cannot be obtained using DMSGETDD.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDD.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDD.

token
(input, CHAR, 8) is a variable for passing the information returned by Open Directory (DMSOPDIR) that uniquely identifies the open directory.

out_dirname_len
(output, INT, 4) is a variable used to return the length of the data passed back in the following *out_dirname* parameter.

out_dirname
(output, CHAR, 153) is a variable in which the name of the directory is passed back. The actual length of the directory name is returned in the preceding *out_dirname_len* parameter.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning the work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request can be asynchronous. CMS returns an integer in *requestid* to identify an asynchronous request. This ID is passed on a later Check (DMSCHECK) request. CMS returns a 1 in *requestid* if the request was completed synchronously and it is not necessary to call DMSCHECK.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates an SFS file control directory or a BFS top directory.

1
indicates an SFS directory control directory.

Usage Notes

1. DMSGETDD returns information about the directory specified on a previous call to the Open Directory (DMSOPDIR) routine. If the open directory is an SFS directory, each subsequent call to DMSGETDD returns information about a subdirectory to which you have authority within the open directory. If the open directory is a BFS top directory, DMSGETDD cannot obtain information about BFS subdirectories. DMSGETDD is supported for use with BFS only for compatibility with existing programs that use the DIR intent.
2. If successful, the requested directory data is returned in the appropriate parameters.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. File pool servers that are not at the current release level do not support all of the DMSGETDD parameters. Rather than return error or warning codes, DMSGETDD tolerates down-level servers where possible. For the unsupported parameters, DMSGETDD returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note “6” on page 230, in the description of DMSGETDI.

Return Codes and Reason Codes

For lists of the possible return codes from DMSGETDD, see Appendix D, “Return Codes,” on page 597.

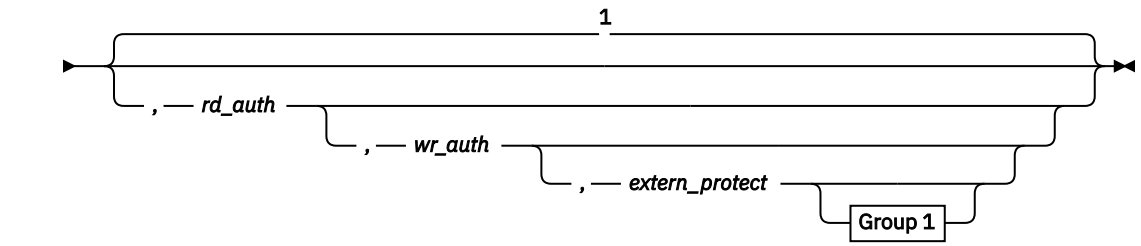
The following table lists the special reason codes returned by DMSGETDD. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

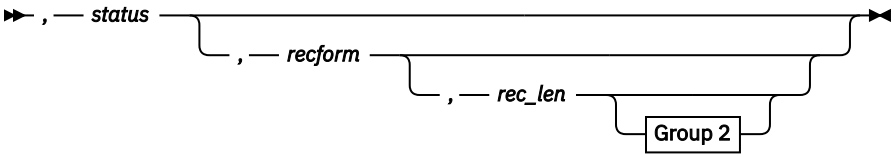
Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of DIR.

Severity	Reason Code	Description
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

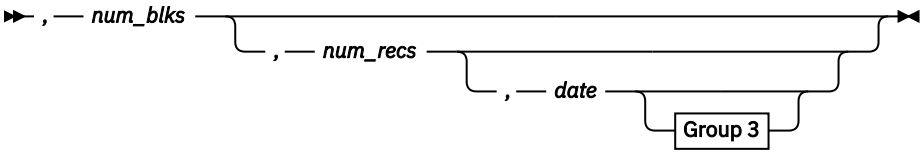
DMSGETDF - Get Directory - File



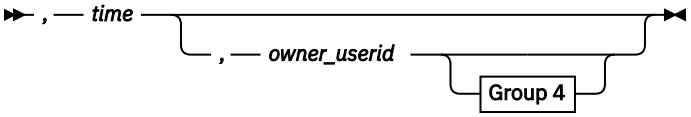
Group 1



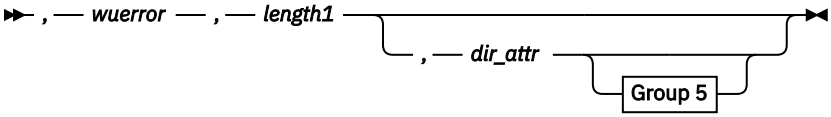
Group 2



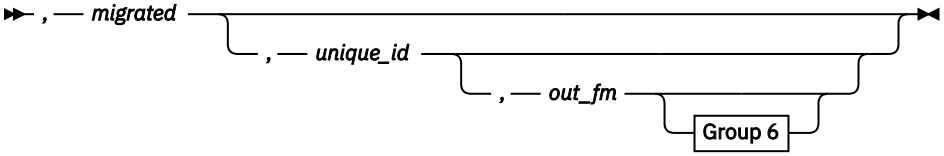
Group 3



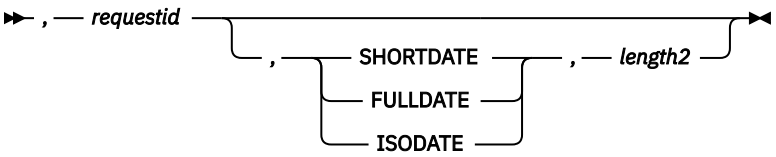
Group 4



Group 5



Group 6



Notes:

[1] Some optional parameters have nonsyntactical default values; see the parameter descriptions.

Context

File System Management: SFS and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSGETDF is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDF
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDF routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of FILE. Immediately after Open Directory, Get Directory starts with the first directory entry. Each subsequent call gets the next directory entry.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDF.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDF.

token
(input, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

fileid
(output, CHAR, 16) is a variable used to return either the file name and file type as two adjacent 8-byte fields, or a subdirectory name.

fm_no
(output, CHAR, 1) is a variable ('0'-'6' or a blank) in which the file mode number is returned.

rd_auth
(output, CHAR, 1) is a 1-byte character in which your read authority is returned:

0
false (always returned for external objects)

1
true

' ' (blank)
check the *extern_protect* indicator

?
authority for the subdirectory is not provided by this function. Use DMSGETDT (or the QUERY AUTHORITY command) to determine the authorization for the object.

wr_auth
(output, CHAR, 1) is a variable in which your write authority is returned:

0
false (always returned for external objects)

1
true (if minidisk file then the file was accessed read/write)

' ' (blank)
check the *extern_protect* indicator

?
authority for the subdirectory is not provided by this function. Use DMSGETDT (or the QUERY AUTHORITY command) to determine the authorization for the object.

extern_protect
(output, CHAR, 1) is a variable in which an indicator is returned to show whether the directory is externally protected; '0' indicates no External Security Manager (ESM) protection; '1' indicates ESM protection.

status
(output, CHAR, 1) is a variable in which the type of the file is returned:

1
base file

2
alias

3
erased alias

4
revoked alias

5
directory

6
external object

7
minidisk file

recform
(output, CHAR, 1) is a variable used to return an indicator to show whether the file contains fixed-length records; variable-length records; a directory; or an erased or revoked alias, or an external object. Possible values are:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

D
indicates that this is a directory record.

- (hyphen)
indicates that the file is an erased or revoked alias, or an external object. See *status* to determine which it is.

rec_len
(output, INT, 4) is a variable in which the record length for the file is returned. A nonzero value is only returned for base files and aliases.

num_blks
(output, INT, 4) is a variable used to return the number of blocks contained in the file. Zero is returned for external objects.

num_recs
(output, INT, 4) is a variable used to return the number of records contained in the file. Zero is returned for external objects.

date
(output, CHAR, 8 or 10) is a variable used to return the date-of-last-update of the file. For external objects, this is the date it was created. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

time
(output, CHAR, 8) is a variable used to return the time the file was last updated or the external object was created, in the character format *hh:mm:ss*.

owner_userid
(output, CHAR, 8) is a variable used to return the user ID of the owner of the base file, directory, or external object. This file contains blanks for a minidisk file.

wuerror
(output, CHAR, *length1*) is a variable used to specify an area for returning work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates a file control directory.

1
indicates a directory control directory.

' ' (blank)
indicates that the returned record does not describe a directory.

migrated
(output, CHAR, 1) is a variable in which a file's migrated status is returned:

0
indicates that file is not in DFSMS/VM migrated status

1
indicates that file is in DFMS/VM migrated status

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

unique_id
(output, CHAR, 16) is a value that uniquely identifies an object within a file pool: base file, alias, alias of an erased file, revoked alias, or external object. It contains blanks for a minidisk file. See usage note "5" on page 226.

Certain administrative actions, such as reorganizing the file pool catalogs, can cause unique ID values to change, but the value for each object is always unique within the file pool.

out_fm
(output, CHAR, 1) is the file mode letter of the accessed minidisk or SFS directory.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request can be asynchronous. CMS returns an integer in *requestid* to identify an asynchronous request. This

ID is passed on a later Check (DMSCHECK) request. CMS returns a 1 in *requestid* if the request was completed synchronously and it is not necessary to call DMSCHECK.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. If the date format parameter is not specified, SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (SHORTDATE, FULLDATE, or ISODATE).

Usage Notes

1. If successful, the requested directory data is returned in the appropriate parameters.
2. DMSGETDF also returns information about subdirectory entries within the directory specified on the call to Open Directory.
3. A file-not-found error condition can occur even though the Open Directory indicated that the file does exist. This can happen if the file is present at the time of the Open Directory but is erased before the Get Directory call is issued.
4. For performance, a question mark (?) is sometimes returned in *rd_auth* and *wr_auth* for subdirectories. The question mark is returned for subdirectories when all of the following are true:
   - The opened (parent) directory is directory control.
   - An External Security Manager is not being used
   - You are not the owner of the subdirectory.

   Use DMSGETDT or the QUERY AUTHORITY command to determine the authorization for these objects.
5. Besides DMSGETDF, CSL routines DMSEXIST, DMSEXIFI, DMSGETDI, and DMSGETDX return the unique ID of an SFS file or directory. You can use DMSEXIST to obtain information about file pool objects identified by the unique ID and file pool ID.
6. File pool servers that are not at the current release level do not support all of the DMSGETDF parameters. Rather than return error or warning codes, DMSGETDF tolerates down-level servers where possible. For the unsupported parameters, DMSGETDF returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note “6” on page 230, in the description of DMSGETDI.
7. If the file resides on a minidisk, the read and write authorities returned indicate how the minidisk was accessed. The *rd_auth* variable will always be true and *wr_auth* will be true if the minidisk was accessed read/write.
8. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
9. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
10. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter,

11. If DMSGETDF is called from a program written in REXX, the *date* field returned will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, this field will be padded on the right with 2 blanks.
12. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.
13. If a DMSGETDF call is made immediately after an ACCESS of a minidisk, the returned directory entries are sorted alphabetically. If the minidisk is updated and not reaccessed, any new directory entries are returned at the end of the list of entries.

Return Codes and Reason Codes

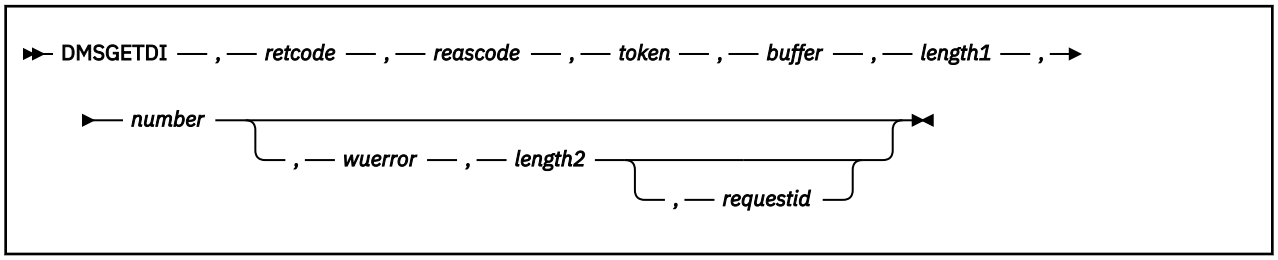
For lists of the possible return codes from DMSGETDF, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSGETDF. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDF can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of FILE.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90310	Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

DMSGETDI - Get Directory



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSGETDI is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDI
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDI routine to read directory records after a directory has been opened using the DMSOPDIR (Open Directory) routine. Immediately after Open Directory, Get Directory starts with either the information for the directory opened or the first directory entry, depending on the intent. Each subsequent call gets the following directory entry. For details of the information returned for each intent, see the description of the related Get Directory routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDI.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDI.

token
(input, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

buffer
(output, CHAR, *length1*) is the area where the directory records read will be returned. The length required for the buffer depends on the intent specified on DMSOPDIR. See Usage Note "17" on page 231 for a description of the record formats and the length of each.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

number
(input/output, INT, 4) is a variable for specifying the number of directory records to read. The number of records must be greater than zero. If the number of records requested cannot be satisfied, this field contains the number of records returned in the buffer. A warning and reason code are returned.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning the work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If, on return, the request ID is still 1, no server call was needed, and it will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. Rather than having the data returned in a buffer, you can use other CSL routines that return information in variables:
   - DMSGETDA (searchall)
   - DMSGETDL (alias)
   - DMSGETDD (dir)
   - DMSGETDS (searchauth)
   - DMSGETDF (file)
   - DMSGETDT (auth)
   - DMSGETDK (lock)
   - DMSGETDX (fileext)

   The assembler mapping macro DIRBUFF is also available to map the output.
2. If DMSGETDI is successful, or partly successful, the requested directory data and the amount of directory data read is returned in *buffer*. The data returned consists of one or more records. The type and length of these records depends on the intent specified on DMSOPDIR. All records are of fixed length and fields that are not used (for example, the requester does not have the authority to receive the information) are set to blanks, if character, or zero, if numeric.
3. DMSGETDI returns as many complete records as possible into the buffer that you provide. (It does not return partial records.) If your buffer is large enough to hold one and a half records, only one record is returned. Use the value in the *length* field in the returned directory record to determine how long a single record is. Use the *number* parameter of the DMSGETDI routine to determine how many records were returned in the buffer.

   If your program was coded prior to this release, it may not have defined a buffer large enough to hold a complete record. DMSGETDI returns fields in groups related to the releases in which they were added to the record, from oldest to newest. This includes any reserved fields used to pad the end of the record for that release. Therefore DMSGETDI returns the fields for the latest release whose record can fit in the specified buffer without truncation. In any case, DMSGETDI gives you the length of the returned record in the length field. To avoid causing a compatibility problem with your program, DMSGETDI does not return a warning reason code for this case.
4. Besides DMSGETDI, CSL routines DMSEXIST, DMSEXIFI, DMSGETDF, and DMSGETDX return the unique ID of an SFS file or directory. You can use DMSEXIST to obtain information about file pool objects identified by the unique ID.

Certain administrative actions, such as reorganizing the file pool catalogs, can cause unique ID values to change, but the value for each object is always unique within the file pool.

5. When you call Get Directory with intent FILE, a file-not-found error condition can occur even though the Open Directory indicated that the file does exist. This can happen if the file is present at the time of the Open Directory but is erased before the Get Directory call is issued.
6. File pool servers that are not at a current release level may not support all file attributes. For a list of these attributes, see usage note "7" on page 194, in the description of DMSEXIST.
7. If a file does not yet have an attribute established, zeros are returned except for:
   - The recoverability attribute, which defaults to a 1 for RECOVER
   - The migration attribute, for which a character '0' is returned
   - The DFSMS/VM related attributes, for which blanks are returned
   - Unique ID attribute, for which blanks are returned.

   For example, if date of last reference is supported but not implemented, the *dec_dateref* field contains 000000 and *dateref* (the character format) contains "00/00/00".
8. The date of last reference, creation date and time, and date and time of last change are recorded in Coordinated Universal Time (UTC). This affects the following fields:
   - *dateref*
   - *dateref_ext*
   - *dec_dateref*
   - *dec_dateref_ext*
   - *iso_dateref_ext*
   - *cr_date*
   - *cr_date_ext*
   - *dec_cr_date*
   - *dec_cr_date_ext*
   - *iso_cr_date_ext*
   - *cr_time*
   - *dec_cr_time*
   - *last_change_date*
   - *last_change_date_ext*
   - *dec_last_change_date*
   - *dec_last_change_date_ext*
   - *iso_last_change_date_ext*
   - *last_change_time*
   - *dec_last_change_time*

   The local date and time of last modification are the local time for the CMS user machine. This affects the following fields:
   - *date*
   - *date_ext*
   - *time*
   - *dec_date*
   - *dec_date_ext*
   - *iso_date_ext*
   - *dec_time*

9. For an alias, the date of last reference (*dateref*, *dateref_ext*, *dec_dateref*, *dec_dateref_ext*, and *iso_dateref_ext* fields) is not updated when either the alias or its base file is referenced. A reference to an alias updates the date of last reference of its base file without altering the date of last reference of the alias. At the time of its creation, an alias acquires the date of last reference of its base file, and this value does not change. For information about using the date of last reference attribute in your program, see the *z/VM: CMS Application Development Guide*.
10. For performance, a question mark (?) is sometimes returned in *rd_auth* and *wr_auth* for subdirectories when the open intent is FILE. The question mark is returned for subdirectories when all of the following are true:
    - The opened parent directory is directory control.
    - An External Security Manager is not being used.
    - You are not the owner of the subdirectory.

    Use DMSGETDT or the QUERY AUTHORITY command to determine the authorization for these objects.
11. The *dir_rd_auth* and *dir_wr_auth* values return your authorization for the directory, not your level of access.
12. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
13. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
14. All minidisk requests are done synchronously.
15. The file ID returned for a BFS file is a system-generated CMS file name and file type.
16. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.
17. The buffer formats for the various types of directory records are:

Table 24. Format of Information Returned for DMSGETDI When Intent Is FILE. (Record Length: 112 bytes)

Offset	Name	Field Type	Description
0 (X'00')	dirtype	char(1)	'1'= FILE
1 (X'01')		char(1)	reserved
2 (X'02')	length	integer(2)	length of the record (including this length field).
4 (X'04')	fileid	char(16)	file identifier; this can be a file name: char(8) and file type: char(8), OR subdirectory: char(16)
20 (X'14')	fm_no	char(1)	file mode number as a character {'0'..'6'}
21 (X'15')	recform	char(1)	record format: 'F'=fixed, 'V'=variable, 'D'=directory, '-'=erased or revoked alias, or external object
22 (X'16')		char(1)	reserved
23 (X'17')	filemode	char(1)	file mode letter if accessed. Blank if not accessed.
24 (X'18')	rec_len	integer(4)	length of the records in the file (0 for directories and external objects)

Table 24. Format of Information Returned for DMSGETDI When Intent Is FILE. (Record Length: 112 bytes) *(continued)*

Offset	Name	Field Type	Description
28 (X'1C')	num_blks	integer(4)	number of blocks in the file (0 for directories and external objects)
32 (X'20')	num_recs	integer(4)	number of records in the file (0 for directories and external objects)
36 (X'24')	dec_date	integer(3)	Local date of last update in decimal format (*yymmdd*)
39 (X'27')	dir_attr	char(1)	directory attribute: ‘0’= file control; ‘1’= directory control; blank if not a directory.
40 (X'28')	dec_time	integer(3)	Local time of last update in decimal format (*hhmmss*).
43 (X'2B')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.
44 (X'2C')	date	char(8)	Local date of last update in character format *yy/mm/dd*
52 (X'34')	time	char(8)	Local time of last update in character format *hh:mm:ss*
60 (X'3C')	owner_userid	char(8)	user ID of owner of base file, external object, or directory. (blank) = minidisk file.
68 (X'44')	status	char(1)	status: ‘1’= base, ‘2’= alias, ‘3’= erased alias, ‘4’= revoked alias, ‘5’= directory, ‘6’= external object, ‘7” = minidisk file
69 (X'45')	rd_auth	char(1)	read authority: ‘0’= false, ‘1’= true, ‘ ’= check the *extern_protect* indicator, ‘?’= authority for the subdirectory not provided by this function (use DMSGETDT or the QUERY AUTHORITY command to get the authorization for this object). For a minidisk file, this always indicates the user has at least read access.
70 (X'46')	wr_auth	char(1)	write authority: ‘0’= false, ‘1’= true, ‘ ’= check the *extern_protect* indicator, ‘?’= authority for the subdirectory not provided by this function (use DMSGETDT or the QUERY AUTHORITY command to get the authorization for this object). If minidisk file, indicates if the minidisk was accessed read/write.
71 (X'47')	extern_protect	char(1)	external protection indicator: ‘0’= none, ‘1’= protected, (blank) = minidisk file

Table 24. Format of Information Returned for DMSGETDI When Intent Is FILE. (Record Length: 112 bytes) *(continued)*

Offset	Name	Field Type	Description
72 (X'48')	unique_id	char(16)	value which is unique for each object (base file, alias, alias of an erased file, directory, revoked alias, or external object) in a file pool. For minidisks, the field contains blanks.
88 (X'58')	dec_date_ext	integer(4)	Local date of last update in decimal format (*yyyymmdd*)
92 (X'5C')	date_ext	char(10)	Local date of last update in character format (*yyyy/mm/dd*)
102 (X'66')	iso_date_ext	char(10)	Local date of last update in character format (*yyyy-mm-dd*)

Table 25. Format of Information Returned for DMSGETDI When Intent Is FILEEXT. (Record Length: 284 bytes)

Offset	Name	Field Type	Description
0 (X'00')	dirtype	char(1)	‘8’= FILEEXT
1 (X'01')	file_space_type	char(1)	‘0’= SFS file space, ‘1’= BFS file space
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	fileid	char(16)	file identifier; for SFS, this can be a file name: char(8) and file type: char(8), OR subdirectory: char(16); for BFS, this is a system-generated file name: char(8) and file type: char(8)
20 (X'14')	fm_no	char(1)	file mode number as a character {‘0’..‘6’}
21 (X'15')	recform	char(1)	record format: ‘F’=fixed, ‘V’=variable, ‘D’=directory, ‘-’=erased or revoked alias, or external object
22 (X'16')	recoverability	char(1)	'1'=RECOVER, '0'=NORECOVER, '-'=N/A
23 (X'17')	overwrite	char(1)	'1'=INPLACE, '0'=NOTINPLACE, '-'=N/A
24 (X'18')	rec_len	integer(4)	length of the records in the file (0 for directories and external objects)
28 (X'1C')	num_blks	integer(4)	number of blocks in the file (0 for directories and external objects)
32 (X'20')	num_recs	integer(4)	number of records in the file (0 for directories and external objects)
36 (X'24')	dec_date	integer(3)	Local date of last update in decimal format (*yymmdd*)
39 (X'27')	dir_attr	char(1)	directory attribute: ‘0’= SFS file control directory or BFS top directory; ‘1’= SFS directory control directory; blank if not a directory.

Table 25. Format of Information Returned for DMSGETDI When Intent Is FILEEXT. (Record Length: 284 bytes) *(continued)*

Offset	Name	Field Type	Description
40 (X'28')	dec_time	integer(3)	Local time of last update in decimal format (*hhmmss*)
43 (X'2B')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.
44 (X'2C')	date	char(8)	Local date of last update in character format *yy/mm/dd*
52 (X'34')	time	char(8)	Local time of last update in character format *hh:mm:ss*
60 (X'3C')	owner_userid	char(8)	user ID that owns base file, external object, or directory
68 (X'44')	status	char(1)	status: ‘1’= SFS base file or BFS regular file, ‘2’= alias, ‘3’= erased alias, ‘4’= revoked alias, ‘5’= directory, ‘6’= external object
69 (X'45')	rd_auth	char(1)	read authority: ‘0’= false, ‘1’= true, ‘ ’= check the *extern_protect* indicator
70 (X'46')	wr_auth	char(1)	write authority: ‘0’= false, ‘1’= true, ‘ ’= check the *extern_protect* indicator.
71 (X'47')	extern_protect	char(1)	external protect indicator: ‘0’= none, ‘1’= protected
72 (X'48')	dec_dateref	integer(3)	UTC date of last reference in decimal format (*yymmdd*)
75 (X'4B')		integer(1)	reserved
76 (X'4C')	dateref	char(8)	UTC date of last reference in character format *yy/mm/dd* (blank for directories and external objects)
84 (X'54')	dec_cr_date	integer(3)	UTC creation date in decimal format *yymmdd*
87 (X'57')		integer(1)	reserved
88 (X'58')	dec_cr_time	integer(3)	UTC creation time in decimal format *hhmmss*
91 (X'5B')		integer(1)	reserved
92 (X'5C')	cr_date	char(8)	UTC creation date in character format *yy/mm/dd*
100 (X'64')	cr_time	char(8)	UTC creation time in character format *hh:mm:ss*

Table 25. Format of Information Returned for DMSGETDI When Intent Is FILEEXT. (Record Length: 284 bytes) *(continued)*

Offset	Name	Field Type	Description
108 (X'6C')	max_blocks	integer(4)	largest block number for the file (0 for directories and external objects)
112 (X'70')	data_blocks	integer(4)	number of blocks used for the file data *only* (0 for directories and external objects)
116 (X'74')	system_blocks	integer(4)	number of blocks used by the system for the file in addition to the data blocks.
120 (X'78')		char(1)	reserved
121 (X'79')	dra_values	char(24)	contains information about DFSMS/VM-related attributes (blank for directories and external objects)
145 (X'91')	unique_id	char(16)	value which is unique for each object (base file, alias, alias of an erased file, directory, revoked alias, or external object) in a file pool
161 (X'A1')	dec_last_change_date	integer(3)	UTC date of last change in decimal format (*yymmdd*)
164 (X'A4')		integer(1)	reserved
165 (X'A5')	dec_last_change_time	integer(3)	UTC time of last change in decimal format (*hhmmss*)
168 (X'A8')		integer(1)	reserved
169 (X'A9')	last_change_date	char(8)	UTC date of last change in character format (*yy/mm/dd*)
177 (X'B1')	last_change_time	char(8)	UTC time of last change in character format (*hh:mm:ss*)
185 (X'B9')	dec_date_ext	integer(4)	Local date of last update in decimal format (*yyyymmdd*)
189 (X'BD')	date_ext	char(10)	Local date of last update in character format (*yyyy/mm/dd*)
199 (X'C7')	iso_date_ext	char(10)	Local date of last update in character format (*yyyy-mm-dd*)
209 (X'D1')	dec_dateref_ext	integer(4)	UTC date of last reference in decimal format (*yyyymmdd*)
213 (X'D5')	dateref_ext	char(10)	UTC date of last reference in character format (*yyyy/mm/dd*). This is blank for directories and external objects.
223 (X'DF')	iso_dateref_ext	char(10)	UTC date of last reference in character format (*yyyy-mm-dd*). This is blank for directories and external objects.
233 (X'E9')	dec_cr_date_ext	integer(4)	UTC creation date in decimal format (*yyyymmdd*).

Table 25. Format of Information Returned for DMSGETDI When Intent Is FILEEXT. (Record Length: 284 bytes) *(continued)*

Offset	Name	Field Type	Description
237 (X'ED')	cr_date_ext	char(10)	UTC creation date in character format (*yyyy/mm/dd*).
247 (X'F7')	iso_cr_date_ext	char(10)	UTC creation date in character format (*yyyy-mm-dd*).
257 (X'101')	dec_last_change_date _ext	integer(4)	UTC date of last change in decimal format (*yyyymmdd*)
261 (X'105')	last_change_date_ext	char(10)	UTC date of last change in character format (*yyyy/mm/dd*)
271 (X'10F')	iso_last_change_date _ext	char(10)	UTC date of last change in character format (*yyyy-mm-dd*)
281 (X'119')		char(3)	reserved

Table 26. Format of Information Returned for DMSGETDI When Intent Is SEARCHALL. (Record Length: 252 bytes)

Offset	Name	Field Type	Description
0 (X'00')	dirtype	char(1)	'2'= SEARCHALL
1 (X'01')		char(1)	reserved
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	fileid	char(16)	file identifier can be a file name: char(8) and file type: char(8), or subdirectory: char(16)
20 (X'14')	fm_no	char(1)	file mode number as a character {'0'..'6', ' '}
21 (X'15')	recform	char(1)	record format: 'F'=fixed, 'V'=variable, 'D'=directory, '-'=erased or revoked alias, or external object
22 (X'16')		char(2)	reserved
24 (X'18')	rec_len	integer	record length (0 for directories and external objects).
28 (X'1C')	num_blks	integer	number of blocks (0 for directories and external objects).
32 (X'20')	num_recs	integer	number of records (0 for directories and external objects).
36 (X'24')	dec_date	integer(3)	Local date in decimal format (*yymmdd*)
39 (X'27')	dir_attr	char(1)	directory attribute: '0'= file control; '1'= directory control; blank if not a directory
40 (X'28')	dec_time	integer(3)	Local time in decimal format (*hhmmss*)

Table 26. Format of Information Returned for DMSGETDI When Intent Is SEARCHALL. (Record Length: 252 bytes) *(continued)*

Offset	Name	Field Type	Description
43 (X'2B')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.
44 (X'2C')	date	char(8)	local date in character format *yy/mm/dd*
52 (X'34')	time	char(8)	local time in character format *hh:mm:ss*
60 (X'3C')	owner_userid	char(8)	user ID that owns base file, external object, or directory.
68 (X'44')	status	char(1)	status: '1'= base, '2'= alias, '3'= erased alias, '4'= revoked alias, '5'= directory, '6'= external object
69 (X'45')		char(3)	reserved
72 (X'48')	dirname_len	integer(1)	directory name length
73 (X'49')	dirname	char(153)	directory name in format: *filepool:userid.subdirs*
226 (X'E2')		char(2)	reserved
228 (X'E4')	dec_date_ext	integer(4)	Local date in decimal format (*yyyymmdd*)
232 (X'E8')	date_ext	char(10)	Local date in character format (*yyyy/mm/dd*)
242 (X'F2')	iso_date_ext	char(10)	Local date in character format (*yyyy-mm-dd*)

Table 27. Format of Information Returned for DMSGETDI When Intent Is SEARCHAUTH. (Record Length: 252 bytes)

Offset	Name	Field Type	Description
0 (X'00')	dirtype	char(1)	'3'= SEARCHAUTH
1 (X'01')		char(1)	reserved
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	fileid	char(16)	file identifier can be a file name: char(8) and file type: char(8), OR subdirectory: char(16)
20 (X'14')	fm_no	char(1)	file mode number as a character {'0'..'6', ' '}
21 (X'15')	recform	char(1)	record format: 'F'=fixed, 'V'=variable, 'D'=directory, '-'=erased or revoked alias
22 (X'16')		char(2)	reserved

Table 27. Format of Information Returned for DMSGETDI When Intent Is SEARCHAUTH. (Record Length: 252 bytes) *(continued)*

Offset	Name	Field Type	Description
24 (X'18')	rec_len	integer	record length (0 for directories)
28 (X'1C')	num_blks	integer	number of blocks (0 for directories)
32 (X'20')	num_recs	integer	number of records (0 for directories)
36 (X'24')	dec_date	integer(3)	Local date in decimal format (*yymmdd*)
39 (X'27')	dir_attr	char(1)	directory attribute: '0'= file control; '1'= directory control; blank if not a directory.
40 (X'28')	dec_time	integer(3)	time in decimal format (*hhmmss*)
43 (X'2B')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.
44 (X'2C')	date	char(8)	Local date in character format *yy/mm/dd*
52 (X'34')	time	char(8)	Local time in character format *hh:mm:ss*
60 (X'3C')	owner_userid	char(8)	user ID that owns base file, external object, or directory.
68 (X'44')	status	char(1)	status: '1'= base, '2'= alias, '3'= erased alias, '4'= revoked alias, '5'= directory.
69 (X'45')	rd_auth	char(1)	read authority: '0'= false, '1'= true, ' '= check the *extern_protect* indicator
70 (X'46')	wr_auth	char(1)	write authority: '0'=false, '1'=true, ' '=check the *extern_protect* indicator
71 (X'47')	extern_protect	char(1)	external protect indicator: '0'= none, '1'= protected
72 (X'48')	dirname_len	integer(1)	directory name length
73 (X'49')	dirname	char(153)	directory name in format: *filepool:userid.subdirs*
226 (X'E2')		char(2)	reserved
228 (X'E4')	dec_date_ext	integer(4)	Local date in decimal format (*yyyymmdd*)
232 (X'E8')	date_ext	char(10)	Local date in character format (*yyyy/mm/dd*)
242 (X'F2')	iso_date_ext	char(10)	Local date in character format (*yyyy-mm-dd*)

Table 28. Format of Information Returned for DMSGETDI When Intent Is ALIAS. (Record Length: 207 bytes)

Offset	Name	Field Type	Description
0 (X'00')	dirtype	char(1)	'4'= ALIAS

Table 28. Format of Information Returned for DMSGETDI When Intent Is ALIAS. (Record Length: 207 bytes) *(continued)*

Offset	Name	Field Type	Description
1 (X'01')		char(1)	reserved
2 (X'02')	length	integer(2)	length of the record (including this length field).
4 (X'04')	fileid1	char(16)	file identifier (matching file ID specified on Open Directory) as file name: char(8) and file type: char(8)
20 (X'14')	fm_no1	char(1)	file mode number as a character {'0'..'6'}
21 (X'15')	status	char(1)	status: '1'= SFS base file or BFS regular file, '2'= alias, '3'= erased alias, '4'= revoked alias
22 (X'16')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.
23 (X'17')		char(1)	reserved
24 (X'18')	num_aliases	integer	number of aliases that exist
28 (X'1C')	dirname_len	integer(1)	directory name length
29 (X'1D')	dirname	char(153)	directory name in format: *filepool:filespace.*[*subdirs*]
182 (X'B6')	fileid2	char(16)	file identifier as file name: char(8) and file type: char(8); if *fileid1* refers to a base file, this is the alias to it, otherwise this is the base file
198 (X'C6')	fm_no2	char(1)	file mode number as a character {'0'..'6'}
199 (X'C7')	userid	char(8)	user ID of the owner of the base file (status= 2, 3, or 4) or the alias (status = 1)

Table 29. Format of Information Returned When Intent Is LOCK. (Record Length: 34 bytes)

Offset	Name	Field Type	Description
0 (X'00')	dirtype	char(1)	'6'= LOCK
1 (X'01')	file_space_type	char(1)	'0'= SFS file space, '1'= BFS file space
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	fileid	char(16)	file identifier; for SFS, this can be a file name: char(8) and file type: char(8), OR subdirectory: char(16); for BFS, this is a system-generated file name: char(8) and file type: char(8)

Table 29. Format of Information Returned When Intent Is LOCK. (Record Length: 34 bytes) *(continued)*			
Offset	**Name**	**Field Type**	**Description**
20 (X'14')	fmnumber	char(1)	file mode number as a character {'0'..'6', ' '}
21 (X'15')	status	char(1)	status: '1'= SFS base file or BFS regular file, '2'= alias, '3'= erased alias, '4'= revoked alias, '5'= directory
22 (X'16')	lock_type	char(1)	lock type indicator: '1'= share, '2'= exclusive, '3'= update
23 (X'17')	lock_len	char(1)	lock length (duration) indicator: '1'= session, '2'= lasting
24 (X'18')	lock_userid	char(8)	user ID of lock owner
32 (X'20')	dir_attr	char(1)	directory attribute: '0'= SFS file control directory or BFS top directory; '1'= SFS directory control directory; blank if not a directory.
33 (X'21')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

Table 30. Format of Information Returned When Intent Is AUTH. (Record Length: 39 bytes)			
Offset	**Name**	**Field Type**	**Description**
0 (X'00')	dirtype	char(1)	'5'= AUTH
1 (X'01')		char(1)	reserved
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	fileid	char(16)	file identifier can be a file name: char(8) and file type: char(8), OR subdirectory: char(16)
20 (X'14')	fmnumber	char(1)	file mode number as a character {'0'..'6'}
21 (X'15')	status	char(1)	status: '1'= base file, '2'= alias, '3'= erased alias, '4'= revoked alias, '5'= directory, '6'= external object
22 (X'16')	rd_auth	char(1)	read authority indicator: '0'= false, '1'= true
23 (X'17')	wr_auth	char(1)	write authority indicator: '0'= false, '1'= true
24 (X'18')	extern_protect	char(1)	external protect indicator: '0'= none, '1'= protected
25 (X'19')	grantee_userid	char(8)	user ID of the user granted the authorities. "*bbbbbbb" for "PUBLIC".

Table 30. Format of Information Returned When Intent Is AUTH. (Record Length: 39 bytes) *(continued)*			
Offset	**Name**	**Field Type**	**Description**
33 (X'21')	dir_rd_auth	char(1)	directory read authority indicator: '0'= false, '1'= true. This field is set only when status=5 (directory)
34 (X'22')	dir_wr_auth	char(1)	directory write authority indicator: '0'= false, '1'= true. This field is set only when status=5 (directory)
35 (X'23')	new_rd_auth	char(1)	NEWREAD authority indicator: '0'= false, '1'= true This field is set only when status=5 (directory)
36 (X'24')	new_wr_auth	char(1)	NEWWRITE authority indicator: '0'= false, '1'= true This field is set only when status=5 (directory)
37 (X'25')	dir_attr	char(1)	directory attribute: '0'= file control; '1'= directory control; blank if not a directory
38 (X'26')	migrated	char(1)	"0" = file is not in DFSMS/VM migrated status; "1" = file is in DFSMS/VM migrated status; (blank) = objects with a *status* value other than 1 (base file) or 2 (alias). A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

Table 31. Format of Information Returned When Intent Is DIR. (Record Length: 159 bytes)			
Offset	**Name**	**Field Type**	**Description**
0 (X'00')	dirtype	char(1)	'7'= DIR
1 (X'01')		char(1)	reserved
2 (X'02')	length	integer(2)	length of the record (including this length field)
4 (X'04')	dirname_len	integer(1)	directory name length
5 (X'05')	dirname	char(153)	directory name in format: *filepool:filespace.*[*subdirs*]
158 (X'9E')	dir_attr	char(1)	directory attribute: '0'= SFS file control directory or BFS top directory; '1'= SFS directory control directory.

18. Output values for BFS objects are as follows:

Table 32. DMSGETDI Output Values for a BFS Object	
Field	**Output Value**
file_space_type	1, indicating a BFS file space
fm_no or *fmnumber*	1, indicating a file mode number of 1
recform	F, indicating fixed-length records
recoverability	1, indicating RECOVER

Table 32. DMSGETDI Output Values for a BFS Object (continued)	
Field	**Output Value**
overwrite	0, indicating NOTINPLACE
rec_len	1, indicating a record length of 1 byte
num_recs	Number of bytes in the file **Note:** If the file has more than 2^{31}-1 bytes, a value of -1 is returned.
dir_attr	0, indicating a BFS top directory
status	1, indicating a BFS regular file
system_blocks	Number of additional blocks generated by the system
owner_userid	Name of the BFS file space

Return Codes and Reason Codes

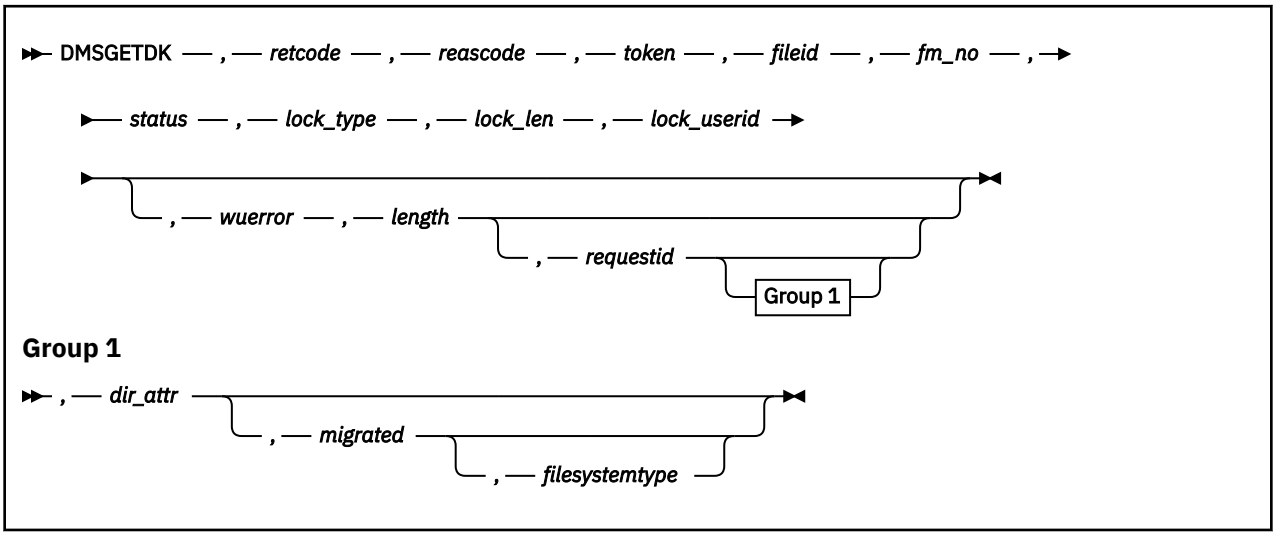
For lists of the possible return codes from DMSGETDI, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSGETDI. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDI can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
WARNING	90280	More data was requested than will fit in the output buffer. There is more data to be returned.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90270	Output buffer is too small to contain one record.
ERROR	90275	No data found for this Get Directory request. For Open Directory LOCK or ALIAS, no locks or aliases were found. For other types of Open Directory, your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90290	Incorrect number of records specified. Number of records must be equal to or greater than one.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

DMSGETDK - Get Directory - Lock



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDK routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of LOCK. Immediately after Open Directory, Get Directory starts with the first directory entry. Each subsequent call gets the next directory entry.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDK.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDK.

token
(input, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

fileid
(output, CHAR, 16) is a variable used to return the file name and file type as two adjacent 8-byte fields. For BFS, these are system-generated values.

fm_no
(output, CHAR, 1) is a variable ('0'-'6' or a blank) in which the file mode number is returned.

status
(output, CHAR, 1) is a variable in which the type of the file is returned:

1
SFS base file or BFS regular file

2
alias

5
directory

lock_type
(output, CHAR, 1) is a variable used to return the type of lock:

1
share

2
exclusive

3
update

lock_len
(output, CHAR, 1) is a variable used to pass back the duration of the lock:

1
means the lock is on for the session

2
means the lock is *lasting* (in effect until a Delete Lock is done).

lock_userid
(output, CHAR, 8) is a variable in which the user ID of the lock owner is returned.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If a 1 is returned, no server call was needed, and it is not necessary to call DMSCHECK because the function has already been completed.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates an SFS file control directory or a BFS top directory.

1
indicates an SFS directory control directory.

' ' (blank)
indicates that the returned record does not describe a directory.

migrated
(output, CHAR, 1) is a variable in which a file's migrated status is returned:

0
indicates the file is not in migrated status.

1
indicates the file is in migrated status.

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

filesystemtype
(output, CHAR, 1) is a variable in which a value is returned that indicates the type of file system:

0
indicates the directory is in an SFS file space.

1
indicates the directory is in a BFS file space.

Usage Notes

1. DMSGETDK returns information about locks on files and subdirectories within the directory specified on the preceding Open Directory call.
2. If DMSGETDK is successful, the requested directory data is returned in the appropriate parameters.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. File pool servers that are not at the current release level do not support all of the DMSGETDK parameters. Rather than return error or warning codes, DMSGETDK tolerates down-level servers where possible. For the unsupported parameters, DMSGETDK returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note "6" on page 230, in the description of DMSGETDI.
6. Output values for BFS objects are as follows:

Table 33. DMSGETDK Output Values for a BFS Object

Parameter	Output Values
fm_no	1, indicating a file mode number of 1
status	1, indicating a BFS regular file
dir_attr	0, indicating a BFS top directory
filesystemtype	1, indicating a BFS file space

Return Codes and Reason Codes

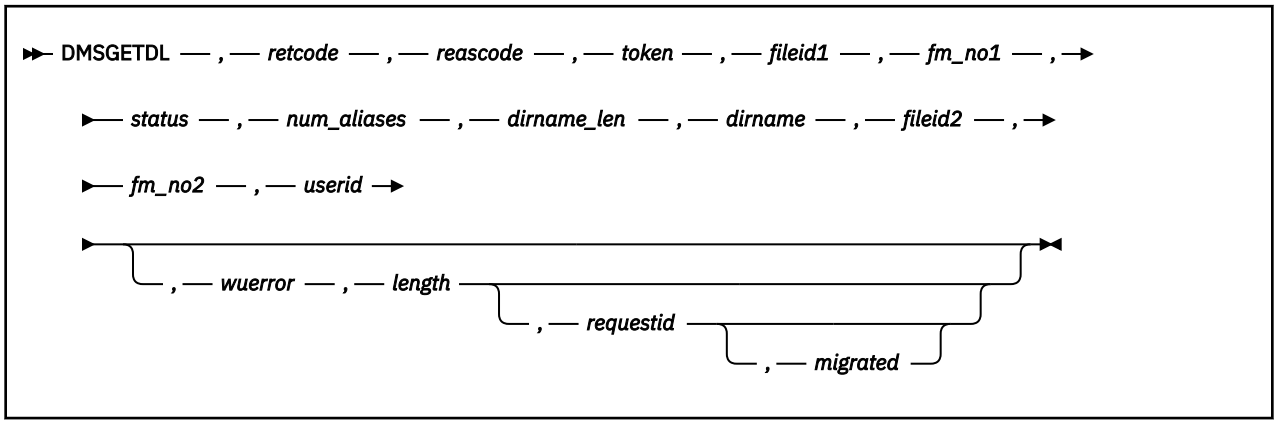
For lists of the possible return codes from DMSGETDK, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSGETDK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of LOCK.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. No locks were found or your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

DMSGETDL - Get Directory - Alias



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDL is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDL
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDL routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of ALIAS. Immediately after Open Directory, Get Directory starts with the first directory entry. Each subsequent call gets the next directory entry.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDL.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDL.

token
(input, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

fileid1
(output, CHAR, 16) is the file name and file type of a file that matches the file ID specified on the Open Directory. It is returned as two adjacent 8-byte fields.

fm_no1
(output, CHAR, 1) is a variable used to return the file mode number ('0' to '6') of the file for which the alias or base file is being requested.

status
(output, CHAR, 1) is a variable in which the type of the file is returned:

1
base

2
alias

3
erased

4
revoked

num_aliases
(output, INT, 4) is a variable used to return the number of aliases that exist.

dirname_len
(output, INT, 4) is a variable used to return the length of the directory name passed back in the following *dirname* parameter. If the value is 0, no directory name is passed back because the issuer is not authorized for the directory.

dirname
(output, CHAR, 153) is a variable used to return the name of the directory containing the alias or base file. The actual length of the directory name is returned in the previous *dirname_len* parameter. If you are not authorized for the directory that contains the alias (if *status*=1) or base file (if *status*=2), this field will be blank. When *status*=3, this field will always be blank because the base file no longer exists.

fileid2
(output, CHAR, 16) is a variable used to return the file ID of the alias (if *status*=1, meaning that *fileid1* specifies a base file) or the base file. The file name and file type are returned as two adjacent 8-byte fields. If you are not authorized for the directory that contains the alias (if *status*=1) or base file (if *status*=2), this field will be blank. When *status*=3, this field will always be blank because the base file no longer exists.

fm_no2
(output, CHAR, 1) is a variable used to return the file mode number ('0'-'6') of the alias or base file. If you are not authorized for the directory that contains the alias (if *status*=1) or base file (if *status*=2), this field will be blank. When *status*=3, this field will always be blank because the base file no longer exists.

userid
(output, CHAR, 8) is a variable used to return the user ID of the owner of the base file if *status* = 2, 3, or 4 (*fileid1* is an alias, revoked alias, or erased alias), or the user ID of the owner of the alias if *status*=1 (*fileid1* is a base file).

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If no server call was needed, 1 is returned. It is not necessary to call DMSCHECK, because the function has already been completed.

migrated
(output, CHAR, 1) is a variable in which a code for a file's migrated status is returned:

0
indicates that file is not in DFSMS/VM migrated status

1
indicates that file is in DFSMS/VM migrated status

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

Usage Notes

1. DMSGETDL returns information about aliases listed within the directory specified on the Open Directory call. If the open directory is a BFS top directory, information cannot be returned about aliases because BFS files do not have aliases. DMSGETDL is supported for use with BFS only for compatibility with existing programs that use the ALIAS intent.
2. A successful request returns the directory information in the appropriate parameters.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. File pool servers that are not at the current release level do not support all of the DMSGETDL parameters. Rather than return error or warning codes, DMSGETDL tolerates down-level servers where possible. For the unsupported parameters, DMSGETDL returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note "6" on page 230, in the description of DMSGETDI.

Return Codes and Reason Codes

For lists of the possible return codes from DMSGETDL, see Appendix D, "Return Codes," on page 597.

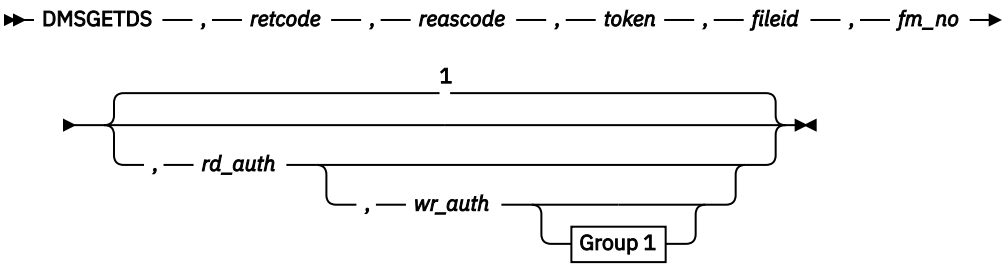
The following table lists the special reason codes returned by DMSGETDL. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDL can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

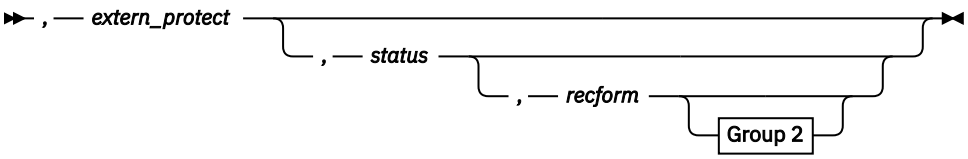
Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of ALIAS.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. No aliases were found, or your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Severity	Reason Code	Description
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

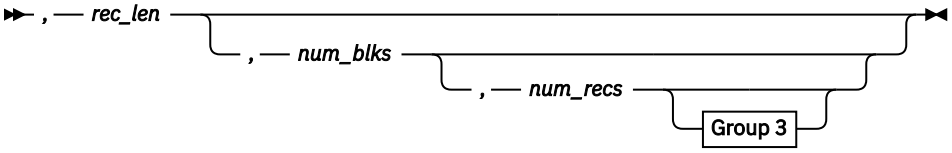
DMSGETDS - Get Directory - Searchauth



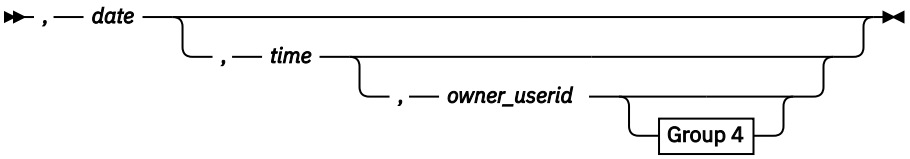
Group 1



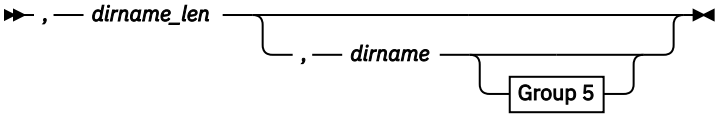
Group 2



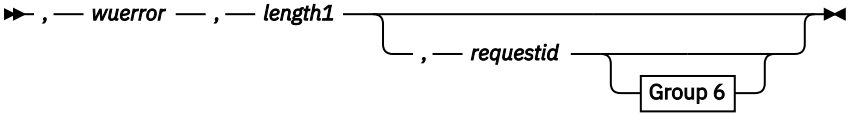
Group 3



Group 4

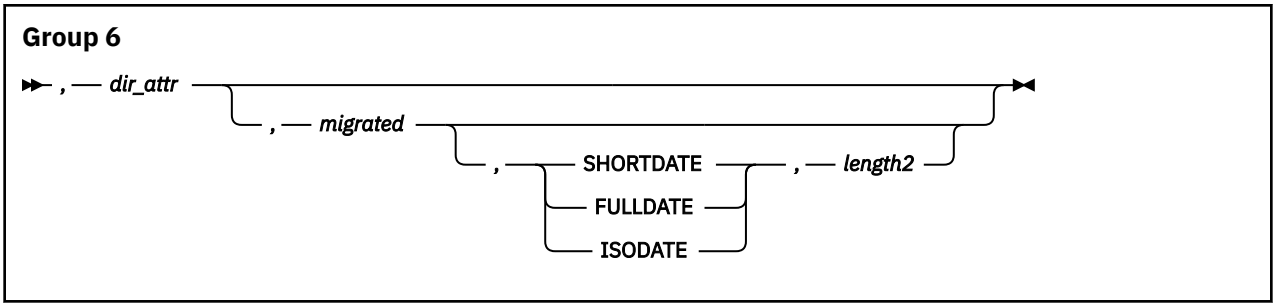


Group 5



Notes:

[1] Some optional parameters have nonsyntactical default values; see the parameter descriptions.



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDS
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDS routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of SEARCHAUTH. Immediately after Open Directory, Get Directory starts with the directory that was opened. Each subsequent call gets the next directory entry within the open directory.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDS.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDS.

token
(output, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

fileid
(output, CHAR, 16) is a variable used to return either the file name and file type as two adjacent 8-byte fields or, a subdirectory name.

fm_no
(output, CHAR, 1) is a variable ('0'-'6' or a blank) in which the file mode number is returned.

rd_auth
(output, CHAR, 1) is a variable in which your read authority is returned:

0
false

1
true

' ' (blank)
check the *extern_protect* indicator

wr_auth
(output, CHAR, 1) is a variable in which your write authority is returned:

0
false

1
true

' ' (blank)
check the *extern_protect* indicator

extern_protect
(output, CHAR, 1) is a variable in which an indicator is returned to show whether the directory is externally protected; '0' indicates no External Security Manager (ESM) protection; '1' indicates ESM protection.

status
(output, CHAR, 1) is a variable in which the type of the file is returned:

1
base

2
alias

3
erased

4
revoked

5
dir

recform
(output, CHAR, 1) is a variable in which an indicator is returned to show whether the file contains fixed-length or variable-length records or an erased or revoked alias. Possible values are:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

D
indicates that this is a directory.

- (hyphen)
indicates that the file is either an erased or revoked alias (see *status* parameter to determine which it is).

rec_len
(output, INT, 4) is a variable in which the record length for the file is returned.

num_blks
(output, INT, 4) is a variable used to return the number of blocks contained in the file.

num_recs
(output, INT, 4) is a variable used to return the number of records contained in the file.

date
(output, CHAR, 8 or 10) is a variable used to return the date the file was created or last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

time
(output, CHAR, 8) is a variable used to return the time the file was created or last updated in the character format *hh:mm:ss*.

owner_userid
(output, CHAR, 8) is a variable used to return the user ID of the owner of the base file or directory.

dirname_len
(output, INT, 4) is a variable used to return the length of the data passed back in the following *dirname* parameter.

dirname
(output, CHAR, 153) is a variable in which the name of the directory containing the file is passed back. The actual length of the directory name is returned in the preceding *dirname_len* parameter.

wuerror
(output, CHAR, *length1*) is a variable used to specify an area for returning work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If 1 is returned, no server call was needed, and it is not necessary to call DMSCHECK because the function has already been completed.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates a file control directory.

1
indicates a directory control directory.

' ' (blank)
indicates that the returned record does not describe a directory.

migrated
(output, CHAR, 1) is a variable in which a file's migrated status is returned:

0
indicates that file is not in DFSMS/VM migrated status

1
indicates that file is in DFMS/VM migrated status

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. If the date format parameter is not specified, SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (SHORTDATE, FULLDATE, or ISODATE).

Usage Notes

1. If you are authorized for the directory specified on the preceding call to the SFS Open Directory (DMSOPDIR) routine, DMSGETDS returns information about that directory and, on subsequent calls, about each entry for which you have authority in that directory, including subdirectories, except those with exclusive locks. When information is returned for the directory that was opened by DMSOPDIR (the first call to DMSGETDA), the *fileid* parameter is left blank.
2. A successful request returns the directory information in the appropriate parameters.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. File pool servers that are not at the current release level do not support all of the DMSGETDS parameters. Rather than return error or warning codes, DMSGETDS tolerates down-level servers where possible. For the unsupported parameters, DMSGETDS returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note "6" on page 230, in the description of DMSGETDI.
6. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
7. If DMSGETDS is called from a program written in REXX, the *date* field returned will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, this field will be padded on the right with 2 blanks.
8. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSGETDS, see Appendix D, "Return Codes," on page 597.

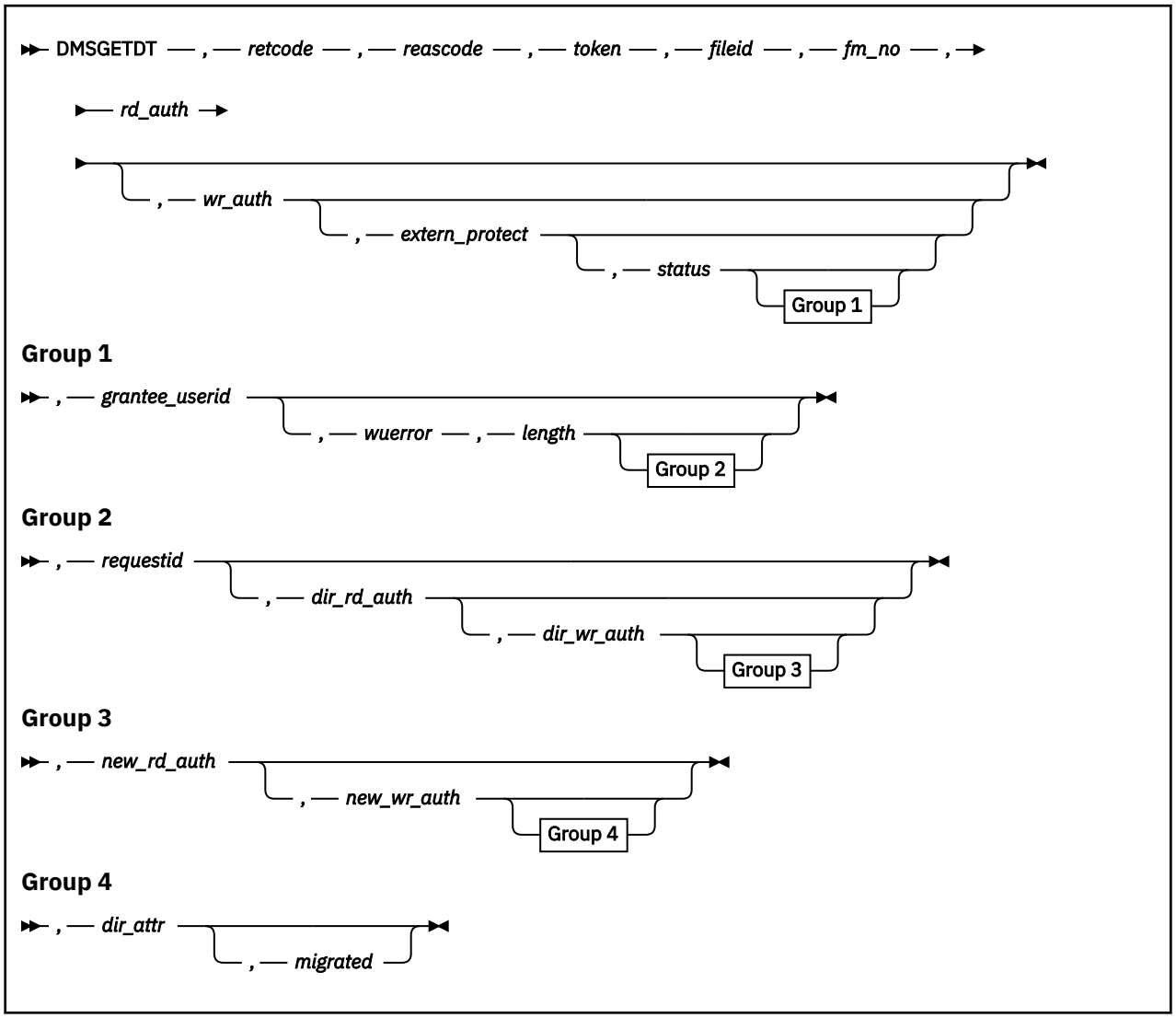
The following table lists the special reason codes returned by DMSGETDS. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDS can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of SEARCHAUTH.

Severity	Reason Code	Description
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90310	Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

DMSGETDT - Get Directory - Auth



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDT routine to read one directory record after a directory has been opened using the DMSOPDIR (Open Directory) routine with an intent of AUTH. Immediately after an Open Directory, Get Directory starts with the first directory entry. Each subsequent call gets the next directory entry.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDT.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDT.

token
(input, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

fileid
(output, CHAR, 16) is a variable used to return either the file name and file type as two adjacent 8-byte fields, or a subdirectory name.

fm_no
(output, CHAR, 1) is a variable in which the file mode number ('0'-'6' or a blank) is returned for the file for which authority information is requested.

rd_auth
(output, CHAR, 1) is a variable used to pass back the read authority of the user ID returned in *grantee_userid*:

0
false (always returned for external objects)

1
true

wr_auth
(output, CHAR, 1) is a variable used to pass back the write authority of the user ID returned in *grantee_userid*:

0
false (always returned for external objects)

1
true

extern_protect
(output, CHAR, 1) is a variable in which an indicator is returned to show whether the directory is externally protected: '0' indicates no External Security Manager (ESM) protection; '1' indicates ESM protection.

status
(output, CHAR, 1) is a variable used to return the type of the file:

1
base file

2
alias

3
erased alias

4
revoked alias

5
directory

6
external object

grantee_userid
(output, CHAR, 8) is a variable used to return the user ID of the user who has been granted the specified authorities to the file or directory indicated. If PUBLIC authority has been granted to the object, "*bbbbbbb" is returned, where "b" is a blank.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If 1 is returned, no server call was needed, and it is not necessary to call DMSCHECK because the function has already been completed.

dir_rd_auth
(output, CHAR, 1) is a variable in which an indicator is returned to show your directory read authority:

0
indicates that you do not have directory read authority.

1
indicates that you have directory read authority.

dir_wr_auth
(output, CHAR, 1) is a variable in which an indicator is returned to show your directory write authority:

0
indicates that you do not have directory write authority.

1
indicates that you have directory write authority.

new_rd_auth
(output, CHAR, 1) is a variable in which an indicator is returned to show your NEWREAD authority:

0
indicates that you do not have NEWREAD authority. This is the value returned for a directory control directory.

1
indicates that you have NEWREAD authority.

new_wr_auth
(output, CHAR, 1) is a variable in which an indicator is returned to show your NEWWRITE authority:

0
indicates that you do not have NEWWRITE authority. This is the value returned for a directory control directory.

1
indicates that you have NEWWRITE authority.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates a file control directory.

1
indicates a directory control directory.

' ' (blank)
indicates that the returned record does not describe a directory.

migrated
(output, CHAR, 1) is a variable in which a file's migrated status is returned:

0
indicates that file is not in DFSMS/VM migrated status

1
indicates that file is in DFMS/VM migrated status

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

Usage Notes

1. DMSGETDT returns authority information about the directory entries for which you have authority within the directory specified on the previous Open Directory call.
2. If DMSGETDT is successful, the requested directory information is returned in the appropriate parameters.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. The *dir_rd_auth* and *dir_wr_auth* parameters represent your actual authority, not your level of access.
6. File pool servers that are not at the current release level do not support all of the DMSGETDT parameters. Rather than return error or warning codes, DMSGETDT tolerates down-level servers where possible. For the unsupported parameters, DMSGETDT returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note “6” on page 230, in the description of DMSGETDI.

Return Codes and Reason Codes

For lists of the possible return codes from DMSGETDT, see Appendix D, “Return Codes,” on page 597.

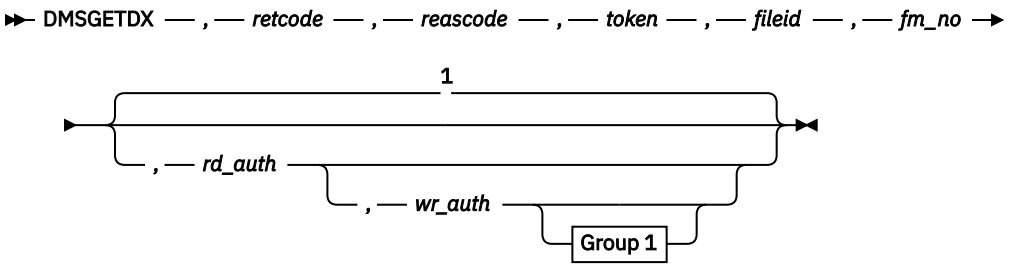
The following table lists the special reason codes returned by DMSGETDT. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

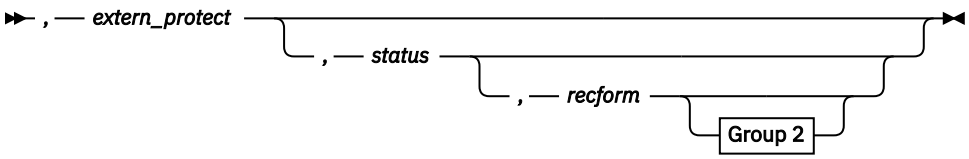
Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of AUTH.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.

Severity	Reason Code	Description
ERROR	90275	No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

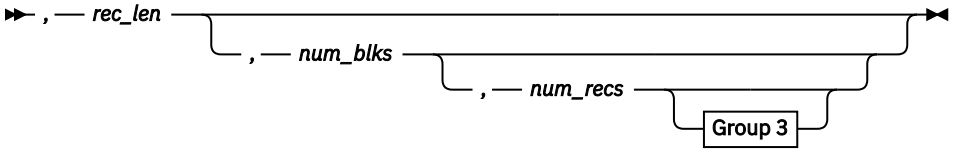
DMSGETDX - Get Directory - File Extended



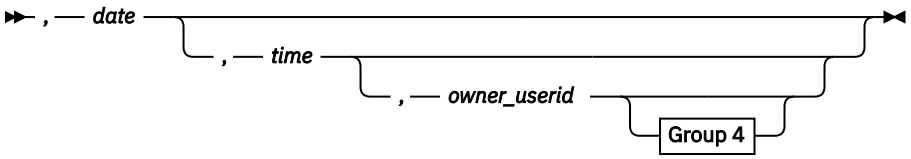
Group 1



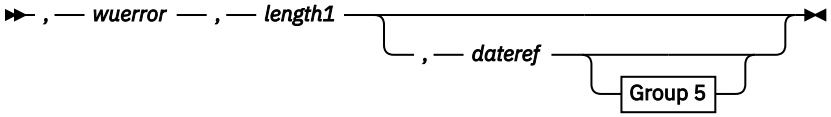
Group 2



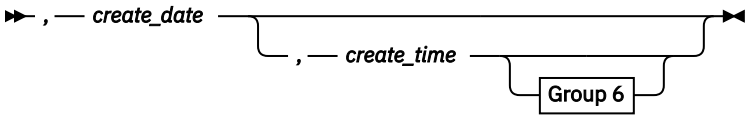
Group 3



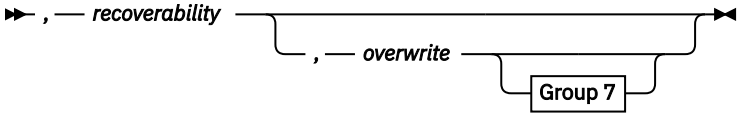
Group 4



Group 5



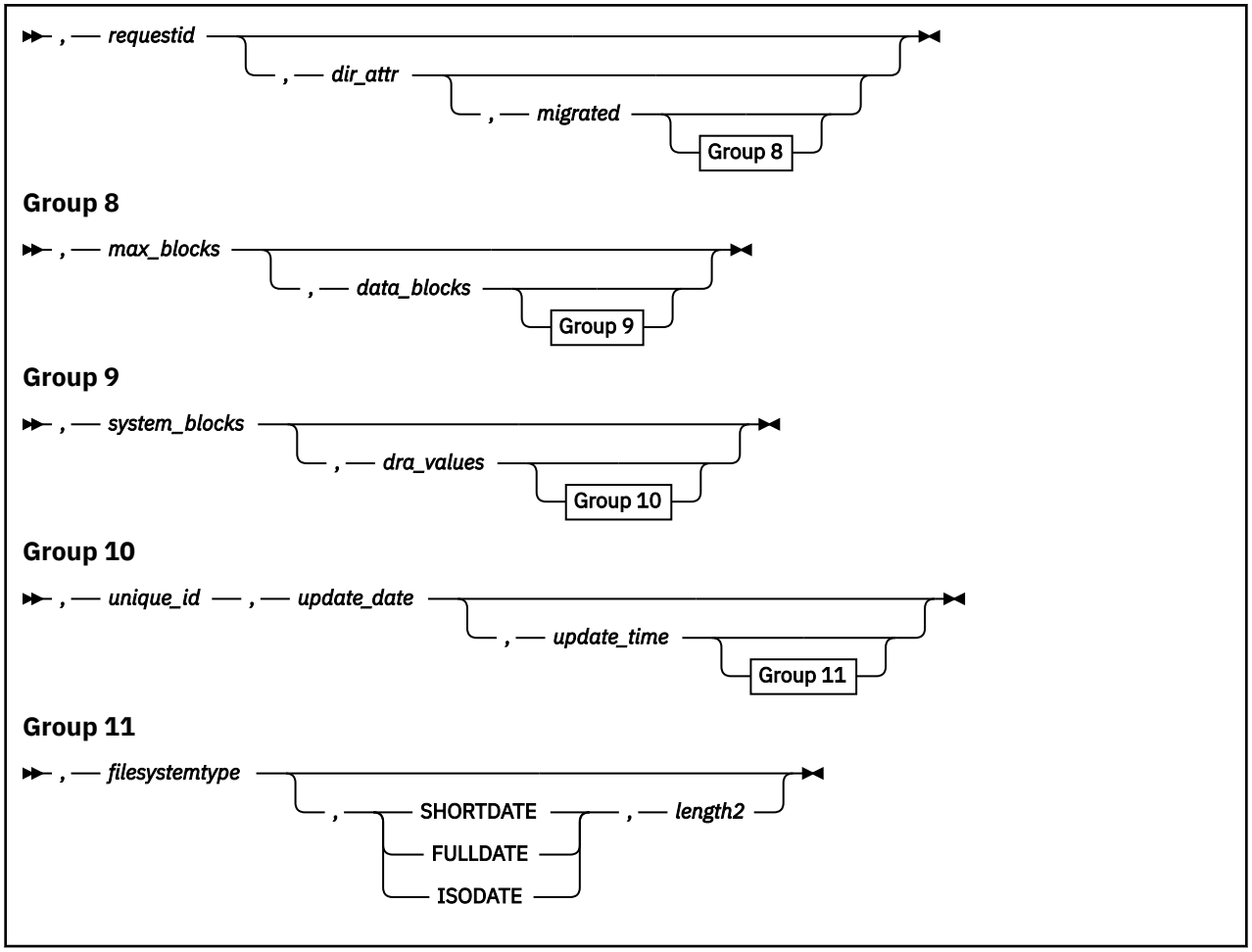
Group 6



Notes:

[1] Some optional parameters have nonsyntactical default values; see the parameter descriptions.

Group 7



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETDX is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETDX
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETDX routine to read a directory record containing extended file attributes. To use DMSGETDX, first open the directory using the DMSOPDIR (Open Directory) routine with an intent of FILEEXT. The first call to DMSGETDX retrieves the first directory entry. The second call retrieves the second entry, and so on.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETDX.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETDX.

token
(input, CHAR, 8) is a variable for passing the information returned by DMSOPDIR that uniquely identifies the open directory.

fileid
(output, CHAR, 16) is a variable used to return either the file name and file type as two adjacent 8-byte fields, or a subdirectory name.

fm_no
(output, CHAR, 1) is a variable ('0'-'6' or a blank) in which the file mode number is returned.

rd_auth
(output, CHAR, 1) is your read authority:

0
false (always returned for external objects)

1
true

' ' (blank)
check *extern_protect* indicator

wr_auth
(output, CHAR, 1) is your write authority:

0
false (always returned for external objects)

1
true

' ' (blank)
check *extern_protect* indicator

extern_protect
(output, CHAR, 1) indicates whether the directory is externally protected: '0' indicates no External Security Manager (ESM) protection, and '1' indicates ESM protection.

status
(output, CHAR, 1) is the type of the file:

1
SFS base file or BFS regular file

2
alias

3
erased alias

4
revoked alias

5
directory

6
external object

recform
(output, CHAR, 1) is a 1-byte character variable used to return an indicator to show whether the file contains fixed-length records; variable-length records; a directory; or an erased or revoked alias, or an external object. Possible values are:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

D
indicates that this is a directory record.

- (hyphen)
indicates that the file is an erased or revoked alias, or an external object. See the *status* parameter to determine which it is.

rec_len
(output, INT, 4) is a variable in which the record length for the file is returned. Zero is returned for directories and external objects.

num_blks
(output, INT, 4) is a variable used to return the number of blocks contained in the file. Zero is returned for directories and external objects.

num_recs
(output, INT, 4) is a variable used to return the number of records contained in the file. Zero is returned for directories and external objects.

date
(output, CHAR, 8 or 10) is a variable used to return the date-of-last-update of the file. For external objects, this is the date it was created. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

time
(output, CHAR, 8) is a variable used to return the time of the last update, in the character format *hh:mm:ss*. For external objects, this is the time it was created.

owner_userid
(output, CHAR, 8) is a variable used to return the user ID of the owner of the base file, directory, or external object.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning work unit extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

dateref
(output, CHAR, 8 or 10) is a variable used to return the date of last reference. The date of last reference is the date the file was created, last read, or last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. Blanks are returned for external objects, and a hyphen followed by 7 or 9 blanks for directories.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

create_date
(output, CHAR, 8 or 10) is a variable used to return the Coordinated Universal Time (UTC) date on which the file or external object was created. The date is returned in a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. A hyphen (-) followed by 7 or 9 blanks is returned for directories.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

create_time
(output, CHAR, 8) is a variable used to return the Coordinated Universal Time (UTC) time at which the file or external object was created. The time is returned in the character format *hh:mm:ss*. A hyphen followed by seven blanks is returned for directories.

recoverability
(output, CHAR, 1) indicates whether a file is to be reset to its last committed state during rollback processing, where:

1
RECOVER

0
NORECOVER

- (hyphen)
Not applicable because file has been erased or revoked, or is a directory or external object. See the *status* parameter to determine which it is.

overwrite
(output, CHAR, 1) indicates whether a reader of the file sees a consistent version of the file throughout the duration of processing:

0
NOTINPLACE

1
INPLACE

- (hyphen)
Not applicable because file has been erased or revoked, or is a directory or external object. See the *status* parameter to determine which it is.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If 1 is returned, no server call was needed, and it is not necessary to call DMSCHECK because the function has already been completed.

dir_attr
(output, CHAR, 1) is a variable in which the directory attribute is returned:

0
indicates an SFS file control directory or a BFS top directory.

1
indicates an SFS directory control directory.

' ' (blank)
indicates that the returned record does not describe a directory.

migrated
(output, CHAR, 1) is a variable in which a file's migrated status is returned:

0
indicates that file is not in DFSMS/VM migrated status

1
indicates that file is in DFMS/VM migrated status

' ' (blank)
is returned for objects with *status* values other than 1 (base file) or 2 (alias).

A file in migrated status is one which has been moved to storage owned and managed by DFSMS/VM.

max_blocks
(output, INT, 4) is a variable that contains the largest block number for the file. Zero is returned for directories and external objects.

data_blocks
(output, INT, 4) is a variable that contains the number of blocks used for the file data *only*. Zero is returned for directories and external objects.

system_blocks
(output, INT, 4) is a variable that contains the number of additional blocks the system needs to use for the file. Zero is returned for directories.

dra_values
(output, CHAR, 24) is a variable that contains three 8-byte fields for DFSMS/VM-related attributes of the file.

unique_id
(output, CHAR, 16) is a value that uniquely identifies an object within a file pool: base file, alias, alias of an erased file, revoked alias, directory, or external object. It contains blanks for a minidisk file. See usage note "4" on page 268.

Certain administrative actions, such as reorganizing the file pool catalogs, can cause unique ID values to change, but the value for each object is always unique within the file pool.

update_date
(output, CHAR, 8 or 10) is the Coordinated Universal Time (UTC) date of the last change to the file, directory, external object, or alias. The date is returned in a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

update_time
(output, CHAR, 8) is the Coordinated Universal Time (UTC) time of the last change to the file, directory, external object, or alias. The format is *hh:mm:ss*.

filesystemtype
(output, CHAR, 1) is a variable in which a value is returned that indicates the type of file system:

0
indicates the object is in an SFS directory.

1
indicates the object is in a BFS directory.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date*, *dateref*, *create_date*, and *update_date* parameters is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. If the date format parameter is not specified, SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date*, *dateref*, *create_date*, and *update_date* parameters is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date*, *dateref*, *create_date*, and *update_date* parameters is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (SHORTDATE, FULLDATE, or ISODATE).

Usage Notes

1. DMSGETDX returns information about directory entries for the directory specified on the previous Open Directory call.
2. If the request is successful, the requested directory information is returned in the appropriate parameters.
3. If you do not need the extended file attributes, do not use this routine. Instead, open the directory with the intent of FILE (not FILEEXT) and use DMSGETDF to read the records. DMSGETDX does not perform as well as DMSGETDF because the extended file attributes are not maintained in your virtual storage, as the regular attributes are. DMSGETDX must communicate with the server to retrieve the attributes.
4. Besides DMSGETDX, CSL routines DMSEXIST, DMSEXIFI, DMSGETDI, and DMSGETDF return the unique ID of an SFS file or directory. You can use DMSEXIST to obtain information about file pool objects identified by the unique ID.
5. If the server is not at the current level, it may not support all the file attributes. See usage note "7" on page 194 for a list of these attributes.

   If the file does not yet have the attribute established for it, character zeros are returned for the *dateref*, *create_date*, *create_time*, and *migrated* parameters; integer zeros for the *max_blocks*, *system_blocks*, and *data_blocks* parameters; and blanks for the *dra_values* and *unique_id* parameters.
6. The date of last reference, creation date and time, and date and time of last change (*dateref*, *create_date*, *create_time*, (*update_date*, and *update_time* fields) are recorded in Coordinated Universal Time (UTC). The date and time of last modification (*date* and *time* fields) are the local time of the CMS user machine.
7. For an alias, the date of last reference (*dateref* field) is not updated when either the alias or its base file is referenced. A reference to an alias updates the date of last reference of its base file without altering the date of last reference of the alias. At the time of its creation, an alias acquires the date of last reference of its base file, and this value does not change. For information about using the date of last reference attribute in your program, see the *z/VM: CMS Application Development Guide*.
8. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
9. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
10. File pool servers that are not at the current release level do not support all of the DMSGETDX parameters. Rather than return error or warning codes, DMSGETDX tolerates down-level servers where possible. For the unsupported parameters, DMSGETDX returns values that are consistent with the functions provided by the down-level server. These values are listed in usage note "6" on page 230, in the description of DMSGETDI.
11. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date*, *dateref*, *create_date*, and *update_date* parameters.

12. If DMSGETDX is called from a program written in REXX, the *date*, *dateref*, *create_date*, and *update_date* fields returned will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, these fields will be padded on the right with 2 blanks.
13. If you want to perform arithmetic or conversion operations on the time stamps that are output from this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.
14. Output values for BFS objects are as follows:

Table 34. DMSGETDX Output Values for a BFS Object

Parameter	Output Value
fm_no	1, indicating a file mode number of 1
status	1, indicating a BFS regular file
recform	F, indicating fixed-length records
rec_len	1, indicating a record length of 1 byte
num_recs	Number of bytes in the file **Note:** If the file has more than 2^{31}-1 bytes, a value of -1 is returned.
recoverability	1, indicating RECOVER
overwrite	0, indicating NOTINPLACE
dir_attr	0, indicating a BFS top directory
system_blocks	Number of additional blocks generated by the system
filesystemtype	1, indicating a BFS file space
owner_userid	Name of the BFS file space

Return Codes and Reason Codes

For lists of the possible return codes from DMSGETDX, see Appendix D, "Return Codes," on page 597.

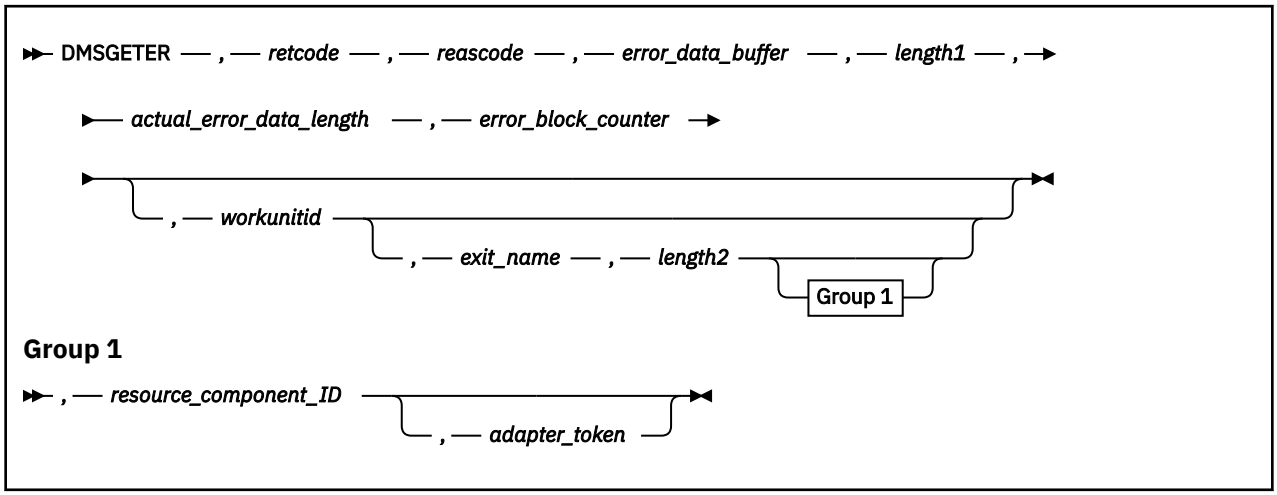
The following table lists the special reason codes returned by DMSGETDX. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETDX can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.
ERROR	90245	Directory was opened for a different intent. It must be opened with intent of FILEEXT.
ERROR	90260	The directory has been closed. The directory was erased, or your authority to it was revoked.
ERROR	90275	No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.
ERROR	90310	Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.

Severity	Reason Code	Description
ERROR	90320	Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	95700	System error. No open directory found for the specified token.

DMSGETER - CRR Get My Errors



Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSGETER is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETER
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETER routine to retrieve, one at a time, error blocks for warnings and errors that a resource adapter has detected since the start of the last commit or backout for the work unit.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETER.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETER.

error_data_buffer
(output, CHAR, *length1*) is a variable for the retrieved error block data.

length1
(input, INT, 4) is a variable for specifying the length (in bytes) of the *error_data_buffer* variable. The *length1* variable defines the maximum amount of data that can be moved. Only the portion of the error block data that fits will be moved into the buffer.

actual_error_data_length
(output, INT, 4) is a variable for the actual length of the error data passed back by the resource adapter's exit routine. This field is not set unless the return code from DMSGETER is 0 or 4.

error_block_counter
(input/output, INT, 4) is a variable for an incremental value that permits the adapter to retrieve all pertinent error blocks by calling DMSGETER repeatedly. The input value on the first call must be 0.

As output (when data is passed back), the value of this parameter identifies the last error block processed. Subsequent calls must return the value *unchanged*. The value is valid until the next sync point.

workunitid
(input, INT, 4) is a variable for the ID of the work unit for which an error block is desired. If the work unit ID is 0 or not specified, the current CMS default work unit is assumed.

exit_name
(input, CHAR, 0–8) is a variable for the name of the resource adapter's exit routine, as specified in the registration (DMSREG) routine.

Only the portion of the name specified in this parameter is used to locate a match in the registry.

length2
(input, INT, 4) is a variable for specifying the length of the *exit_name* variable.

If the *exit_name* parameter has been set to 0 (null), the *length2* parameter must also be 0. Specifying 0 has the same effect as omitting the *exit_name* parameter.

resource_component_id
(input, CHAR, 9) is a variable for the component identification for the resource whose error blocks are being retrieved. If used as a search argument, this ID must match the ID specified in the registration (DMSREG) routine. A blank in the first byte means this parameter is not to be used as a search argument.

adapter_token
(output, CHAR, 4) is a variable that identifies the resource associated with the error block. This value was supplied by the resource adapter in the registration (DMSREG) routine.

Usage Notes

1. For guidance information on using the DMSGETER routine in the context of getting a resource manager to participate in CRR, see *z/VM: CMS Application Development Guide*.
2. DMSGETER should be called repeatedly to retrieve all error blocks until the routine returns a return code of 4 and a reason code of 44040 (all data returned).
3. Getting an error block is nondestructive. Two programs (such as the application and the resource adapter) could get the errors for the same work unit, component, and exit name at the same time but not interfere with each other.
4. Combinations of the *workunitid* parameter, *exit_name* parameter, and *resource_component_id* parameter can be used to set search criteria for error blocks. All the specified search arguments must be met before an error block is returned to the caller.
5. The *adapter_token* parameter is intended for use by the resource manager's routine that passes back resource-specific error information, and should enable the resource adapter to identify the resource that had the error. Only the resource adapter keeps resource identification data, not the SPM.
6. Error data is preserved until the start of the next synchronization point or until the work unit ends. A call to DMSUNREG to unregister the resource does not delete its error data.

Return Codes and Reason Codes

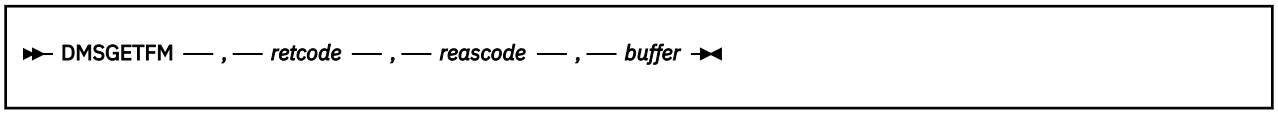
For lists of the possible return codes from DMSGETER, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSGETER. WARNING means the request was processed, but a warning condition was encountered; ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8.

DMSGETER can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	All data has previously been returned. The error data buffer is unchanged.
WARNING	90271	All available information would not fit into the error data buffer. As much data as would fit into the buffer has been placed there. Check the *actual_error_data_length* parameter to find out the actual length of the data in the error block. To retrieve all the available data, make the buffer equal to the returned length, set the *error_block_counter* parameter to 0, and call DMSGETER again.
WARNING	90278	No error blocks were found. Either there were no sync point warnings or errors for the work unit, or the participant that received the warnings or errors did not create any error blocks.
ERROR	90118	Invalid *error_block_counter* parameter. The caller changed the counter, or a sync point occurred since the call to DMSGETER that created the counter.
ERROR	90210	Extraneous characters in input parameter.
ERROR	90277	No matching registered resource was found for the input parameters you specified.
ERROR	90300	Invalid input parameter.
ERROR	90485	Invalid buffer length specified.
ERROR	90540	Invalid *workunitid* parameter.

DMSGETFM - Get File Mode



Call Format

The format for calling a CSL routine is language dependent. DMSGETFM is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETFM
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETFM routine to find the first unaccessed file mode letter. The search starts with file mode A.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETFM.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETFM.

buffer
(output, CHAR, 1) is a variable containing, upon return, the first unaccessed file mode letter.

Return Codes and Reason Codes

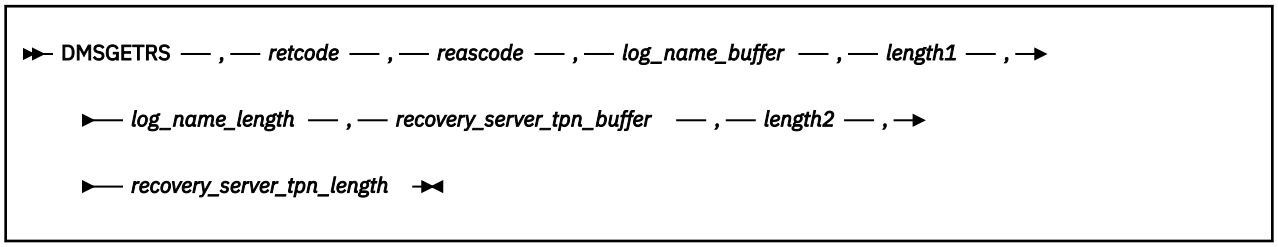
For lists of the possible return codes from DMSGETFM, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSGETFM. ERROR means the request failed. Errors cause return code 8 or 12.

DMSGETFM can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	90600	No free file modes available.

DMSGETRS - CRR Get Recovery Server Information



Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSGETRS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETRS
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETRS routine to get the CRR recovery server's current log name and transaction program name (TPN). The resource adapter calls this routine to get the recovery server information it must provide to its resource manager. The resource manager then uses the information to allocate an APPC conversation with the CRR recovery server.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETRS.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETRS.

log_name_buffer
(output, CHAR, 16–64) is a variable for the CRR recovery server's log name.

length1
(input, INT, 4) is a variable for specifying the length (in bytes) of the *log_name_buffer* variable.

log_name_length
(output, INT, 4) is a variable for the length of the CRR recovery server's log name.

recovery_server_tpn_buffer
(output, CHAR, 8–24) is a variable for the CRR recovery server's TPN.

length2
(input, INT, 4) is a variable for specifying the length (in bytes) of the *recovery_server_tpn_buffer* variable.

recovery_server_tpn_length
(output, INT, 4) is a variable for the length of the CRR recovery server's TPN.

Usage Notes

1. For guidance information on using the DMSGETRS routine in the context of getting a resource manager to participate in CRR, see *z/VM: CMS Application Development Guide*.
2. If the resource manager is on the same system as the resource adapter, the resource manager can call DMSGETRS.

Return Codes and Reason Codes

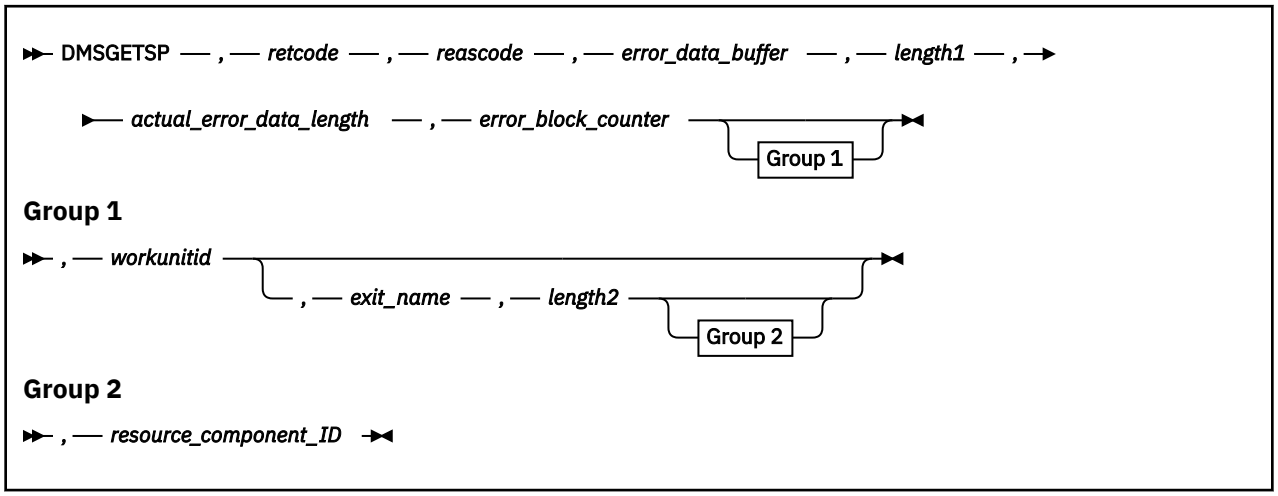
For lists of the possible return codes from DMSGETRS, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSGETRS. ERROR means the request failed. Errors cause return code 8.

DMSGETRS can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	55000	Insufficient virtual storage in the CRR recovery server.
ERROR	75000	The service levels of the CRR recovery server and the user machine are not compatible.
ERROR	81056	The CRR recovery server is unavailable.
ERROR	90485	Incorrect buffer length specified.
ERROR	95200	System error. Further attempts to access a CRR recovery server will be rejected.
ERROR	97280	Your attempt exceeds the number of APPC/VM connections allowed for the CRR recovery server.

DMSGETSP - Get Synchronization Point Errors



Context

Error Checking and Debugging

Call Format

The format for calling a CSL routine is language dependent. DMSGETSP is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETSP
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETSP routine to iteratively retrieve all error blocks for all warnings and errors detected since the start of the last commit or rollback for a work unit. This routine is especially useful if your application writes to multiple file pools or other resources, or uses a protected conversation. If your application receives an indication of a data integrity problem (return code of 16 or 20), this routine can help determine which resource experienced the problem.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETSP.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETSP.

error_data_buffer
(output, CHAR, *length1*) is a variable for specifying a buffer area to receive an error block's data. The following *length1* parameter specifies the size of this area.

length1
(input, INT, 4) is a variable for specifying the size of the *error_data_buffer* area. This is the maximum amount of data that can be received. Only that portion that fits will be moved into the caller's data area (*error_data_buffer*).

actual_error_data_length
(output, INT, 4) is a variable for returning the actual length of the error data in the resource's error block. If the value returned is greater than the length of the caller's data area (*error_data_buffer*), then all available data was not returned. This field is filled in only when this routine executes successfully (return code is 4 or less).

error_block_counter
(input/output, INT, 4) specifies a variable to act as a place holder or cursor, letting your application receive all pertinent error blocks by repeatedly calling Get Synchronization Point Errors. The input value on a first call must be zero in order to start at the first error block for this work unit. The value that is passed back helps DMSGETSP keep track of which error blocks have been processed. Your application must not change this value on subsequent calls. A given *error_block_counter* value is valid until the next synchronization point (commit or rollback).

workunitid
(input, INT, 4) is a variable in which you can specify the work unit for which error block data is desired. If you do not specify it, or specify it as zero, the current default work unit is assumed.

exit_name
(output, CHAR, 0-8) is a variable for returning the CSL routine name of the exit that filled in this error block. Your application can use this field to identify the resource type for the resource that had an error. Your application should set the following *length2* field to 8 so that a maximum size exit name will fit in the output field. In any case, only the part of the name that fits (maximum of 8 bytes) is returned. The field is padded on the right with blanks if necessary.

length2
(input, INT, 4) specifies the length of the *exit_name* parameter. Specifying a length of zero has the effect of omitting the *exit_name* parameter (the exit name is not filled into the variable). If the length is specified as 0, the *exit_name* parameter must be 0 (meaning null).

resource_component_ID
(output, CHAR, 9) returns the resource type for the resource that had an error. The 9-byte component ID should be provided in the resource documentation. For IBM products, the component ID can be found in the *Programming System General Information Manual*.

Usage Notes

1. The application should call DMSGETSP repeatedly until a warning return code of 4 and a reason code of 44040 are returned indicating that all data has been returned. For an example of how DMSGETSP can be used, see the *z/VM: CMS Application Development Guide*.
2. If a return code of 4 and a reason code of 90271 is returned, check the *actual_error_data_length* parameter for the actual length of the error data in the resource's error block. To get all the data for an error block into the buffer, provide an *error_data_buffer* buffer of the size indicated by the *actual_error_data_length* parameter, specify that length in the *length1* parameter, set the *error_block_counter* to 0, and reissue the call to DMSGETSP.
3. You can use the *exit_name* and *resource_component_ID* parameters to uniquely identify the resource manager and then check the product's documentation for the format of the error data. The documentation for the resource manager may provide either or both of these in its product manuals. For example, the resource component ID for CMS is 5749DMS00; the SFS exit name is DMS2AE and the APPC protected conversation exit name is DMSPCA.
4. The format of SFS error data can be found in the description of the *wuerror* parameter, see "Common Parameters" on page 15.
5. The format of CMS APPC protected conversation error data is described in the DMSPCAER CSL routine.
6. Getting the contents of an error block is nondestructive, so two programs can get the error information for the same work unit simultaneously without interfering with each other.
7. Error data for a synchronization point is preserved until the start of the next synchronization point or until the work unit ends.
8. Even though the return code on a synchronization point is zero, error blocks may be created due to errors encountered during the synchronization point.

Return Codes and Reason Codes

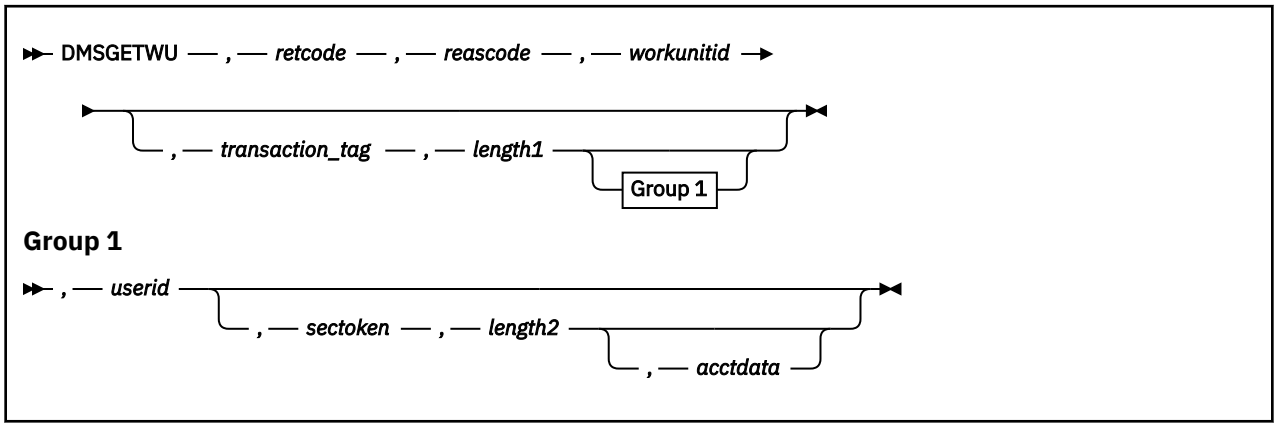
For lists of the possible return codes from DMSGETSP, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSGETSP. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSGETSP can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	44040	All data has previously been returned. The *error_data_buffer* buffer is unchanged.
WARNING	90271	Warning, all available information would not fit in the *error_data_buffer* buffer. As much data as would fit in the buffer has been placed there. Check the *actual_error_data_length* parameter to get the actual length of the usable data.
WARNING	90278	No error blocks were found. This means that there were no synchronization point errors for this work unit, or that the protected resource did not create any error blocks.
ERROR	90118	Invalid value in the *error_block_counter* parameter. The caller changed the value in the *error_block_counter* parameter or a synchronization point occurred since the call to DMSGETSP that created it.
ERROR	90485	Invalid buffer length specified.
ERROR	90540	Specified work unit ID is invalid.

DMSGETWU - Get Work Unit ID



Context

Work Unit Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSGETWU is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGETWU
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGETWU routine to obtain a work unit ID that is unique within a virtual machine. If no unique work unit IDs are available, an error is returned. You can also provide a default log message for all transactions (units of work) done for this work unit.

DMSGETWU can also be used by a multiuser source server to associate a work unit with a user ID originating a file pool request.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGETWU.

reascode
(output, INT, 4) is a variable for the reason code from DMSGETWU.

workunitid
(output, INT, 4) is a variable used for returning the work unit identifier.

transaction_tag
(input, CHAR, 1-80) is a variable for specifying up to 80 characters of information about the transaction being processed. This information should enable the recovery operator (local CRR recovery server operator or local resource manager operator) to determine what needs to be done in case of a problem requiring operator intervention. The information supplied is displayed by the CRR QUERY LU or QUERY LUWID command. SFS also supports transaction tags; the QUERY PREPARED command displays the information provided.

Transaction tags are also displayed in some resynchronization messages.

length1
(input, INT, 4) is a variable for specifying the length, up to 80 bytes, of the preceding character parameter (*transaction_tag*). Specify 0 to omit this parameter.

userid
(input, CHAR, 8) is the user ID you wish to associate with file pool requests for this work unit. File pool administration authority is required to use this parameter; specify binary zeros to skip this parameter. If a null value (8 bytes of binary zeros) is specified for *userid*, it is assumed to be the user ID of the user machine, and no values of *sectoken* or *acctdata* are associated with the work unit.

sectoken
(input, CHAR, 0-64) is the security token you wish to associate with this work unit.

length2
(input, INT, 4) is the length of *sectoken* in bytes. Specify 0 to skip the *sectoken* parameter.

acctdata
(input, CHAR, 16) is accounting data associated with *userid*. Specify 16 bytes of binary zeros (the null value) to skip this parameter.

Usage Notes

1. The application can change the transaction tag by calling the routine described in "DMSSETAG - Set Transaction Tag" on page 480. If there are special recovery requirements for a particular synchronization point, then Set Transaction Tag should be issued before that synchronization point occurs.
2. You can associate a user ID with a work unit at the time you assign the work unit ID by specifying the *userid*, *sectoken*, and *acctdata* parameters on the request along with the work unit ID.

   The *userid* parameter lets a service machine execute a request on behalf of another user. The *sectoken* parameter is passed to an External Security Manager and lets a service machine execute a request from another machine that has logged off since making the request. The *acctdata* parameter lets a service machine store accounting data for other user IDs than its own.
3. Attempts to obtain a session lock on an SFS or BFS resource for *userid* are rejected with an error reason code. (See routine, "DMSCRLOC - Create Lock" on page 102 for more information on the SESSION parameter.) You can create a *lasting* lock for another user.
4. When you associate another user ID with a work unit, you must have administration authority for the file pools you use (see the discussion of multiple user ID support in the *z/VM: CMS Application Development Guide*). Lack of administration authority is detected when the first connection is made to the file pool and error reason code 30000 is returned.
5. Take these precautions when you obtain a work unit for another user ID, or the user ID can be given access to objects it is not authorized for or denied access to objects it is authorized for:
   - Use CSL routines to access file pools rather than CMS commands or the compatibility interfaces. You can specify the work unit on the calls to the routines and make sure that the correct user ID is associated with the work being done. With CMS commands and the compatibility interfaces, you do not have full control over which work unit is used.
   - On a call to a CSL routine, specify the user ID associated with the work unit in the directory name or in a namedef for the directory name. The default user ID in directory name is the user ID of the virtual machine in which the routine is running, not the user ID for which the work unit was obtained.
   - Do not use any accessed directories on the work unit.
6. A security token and accounting user data can be associated with a work unit ID *only* if a non-null user ID is also specified.
7. If the file pool is managed by an external security manager and *userid* is specified without *sectoken*, or with an incorrect *sectoken*, DMSGETWU does not return an error, but subsequent authorization checks by the external security manager may fail.
8. If you call DMSGETWU with the user ID parameter (with or without the *sectoken* and *acctdata* parameters), call DMSRETWU or DMSPURWU to return the work unit after processing is complete.

This ensures that returned and purged work units are no longer associated with a user ID, security token, or user accounting data.

9. Specifying the user ID parameter causes an error if you try to connect to a file pool managed by a file pool server at the z/VM Version 1 Release 1.0 or earlier level.

Return Codes and Reason Codes

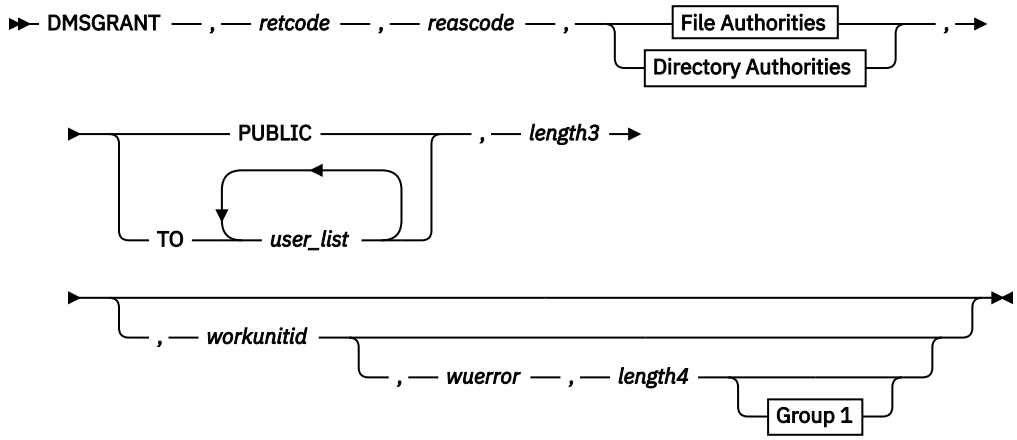
For lists of the possible return codes from DMSGETWU, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSGETWU. ERROR means the request failed. Errors cause return code 8 or 12.

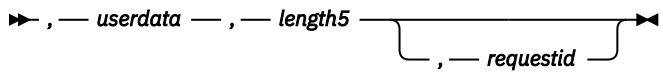
DMSGETWU can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90300	Invalid input parameter in CSL parameter list. The specified transaction tag length is invalid.
ERROR	90410	Invalid parameter length specified (*sectoken* can be 0-64, and *transaction_tag* can be 0-80 characters).
ERROR	90540	Invalid work unit ID.
ERROR	90550	No work unit IDs are available. You must re-IPL your virtual machine to make some work unit IDs available.
ERROR	96610	CSL error calling DMSSETAG Set Transaction Tag. Routine dropped or not loaded.

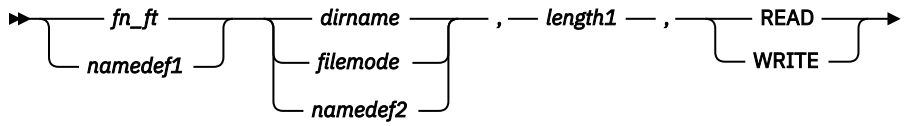
DMSGRANT - Grant Authority



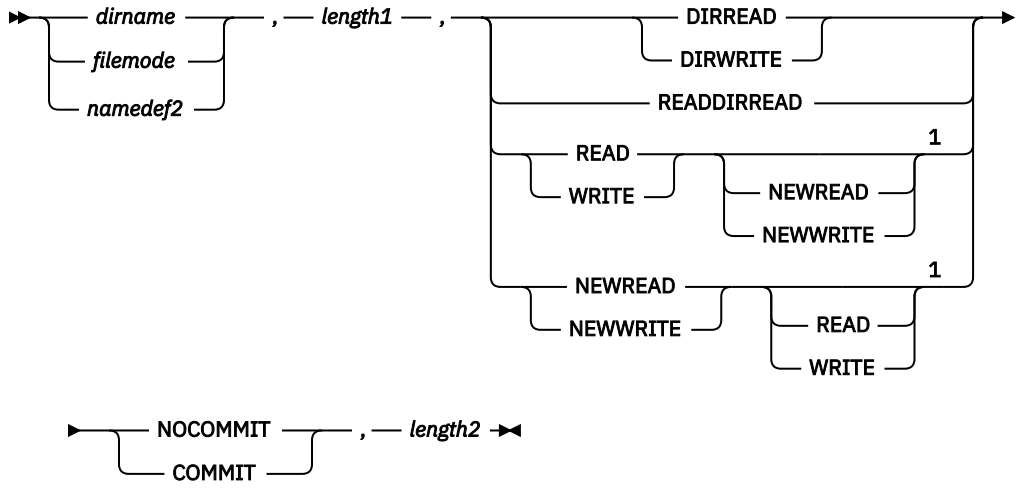
Group 1



File Authorities



Directory Authorities



Notes:

[1] For a file control directory, you can grant one or two authorities, in any order. See usage note "1" on page 286.

Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSGRANT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSGRANT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSGRANT routine to give another user authority to SFS files and directories that you own. It cannot be used for files and directories protected by an External Security Manager (ESM).

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSGRANT.

reascode
(output, INT, 4) is a variable for the reason code from DMSGRANT.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file for which authority is being granted.

namedef1
(input, CHAR, 1-16) is a variable for specifying a namedef (a temporary name) of the file for which authority is being granted.

dirname
(input, CHAR, 1-153) is a variable for specifying the specified directory in which the file resides or to which authority is being granted. If *fn_ft* or *namedef1* is specified, this is the directory containing the file to which authority is being granted. If *fn_ft* or *namedef1* is not specified, this is the directory to which authority is being granted.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If this file mode letter is also a one character namedef (*namedef2*), the namedef will be used. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable for specifying a namedef (a temporary name) for a directory. If *fn_ft* or *namedef1* is specified, this is the directory containing the file to which authority is being granted. If *fn_ft* or *namedef1* is not specified, this is the directory to which authority is being granted.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, if specified, plus *dirname*, *filemode*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

READ
(input, CHAR, 4) specifies READ authority. READ authority on a file lets the user read the contents of a file.

READ authority on a directory lets the user read the directory contents, which consists of the names of all files and subdirectories. It also allows use of the ACCESS command to access the directory. READ authority applies only to file control directories and files in them.

WRITE
(input, CHAR, 5) specifies WRITE authority. WRITE authority on a file implies both READ and WRITE authority. It means a user can read, modify, or erase a file. To erase a file, a user must also have write authority on the directory in which the file resides.

WRITE authority on a directory gives the user read authority and the authority to create files and aliases in that directory. It does not give authorization to read or write to any of the files in that directory.

WRITE authority applies only to file control directories and the files within them.

NEWREAD
(input, CHAR, 7) indicates that the users will automatically receive READ authority for any new base files added to the directory. When NEWREAD authority is granted to a user, authorizations to existing files in the directory are not affected. NEWREAD authority does not give access to aliases that are added to the directory, nor to their base files.

The NEWREAD option is valid only for file control directories. (The *fn_ft* and *namedef1* parameters must be omitted.)

NEWWRITE
(input, CHAR, 8) indicates that the users will automatically receive WRITE authority for any new base files added to the directory. When NEWWRITE authority is granted to a user, authorizations to existing files in the directory are not affected. NEWWRITE authority does not give access to aliases that are added to the directory, nor to their base files.

The NEWWRITE option is valid only for file control directories. (The *fn_ft* and *namedef1* parameters must be omitted.)

DIRREAD
(input, CHAR, 7) indicates that the users will receive directory read (DIRREAD) authority on the specified directory. DIRREAD authority lets the user:

- Read the directory
- Read files currently in the directory
- Read any files added to the directory in the future
- Access the directory read-only.

DIRREAD authority can be granted only on directory control directories. Omit both *fn_ft* and *namedef1* when granting DIRREAD authority.

DIRWRITE
(input, CHAR, 8) indicates that the users will receive directory write (DIRWRITE) authority on the specified directory. DIRWRITE authority lets the user:

- Read from and write to the directory
- Read from and write to any files currently in the directory
- Read from and write to any files added to the directory in the future
- Access the directory read-only or read/write.

DIRWRITE authority can be granted only on directory control directories. Omit both *fn_ft* and *namedef1* when granting DIRWRITE authority.

READDIRREAD
(input, CHAR, 11) indicates that the user should be granted either READ or DIRREAD authority. READDIRREAD is valid only when granting authority on a directory. READ authority is granted if the specified directory is file control. DIRREAD authority is granted if the specified directory is directory control.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under “Common Parameters” on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (READ or WRITE or NEWREAD or NEWWRITE or DIRREAD or DIRWRITE or READDIRREAD, and COMMIT or NOCOMMIT). See "Compound Variables" on page 15 for coding details.

PUBLIC
(input, CHAR, 6) indicates that authority is granted to all users in the system who can connect to the file pool.

TO *user_list*
(input, CHAR, *length3*) indicates that authority is granted to the user IDs specified. User IDs must be separated by blanks, and user IDs cannot begin with a plus (+) or a minus (–) or contain a colon (:) or a period (.). Nicknames are not allowed.

length3
(input, INT, 4) is a variable containing the length of the preceding character parameter (PUBLIC, TO *user_list*).

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length5* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. The authorities you can grant and revoke are:
   - For SFS file control directories:
     - READ authority
     - READDIRREAD, which is the same as granting READ authority
     - WRITE authority (which implies READ authority)
     - NEWREAD authority
     - NEWWRITE authority (which implies NEWREAD authority).

     NEWREAD and NEWWRITE are independent of READ and WRITE.

- For files within file control directories:
  - READ authority
  - WRITE authority (which implies READ authority).
- For SFS directory control directories:
  - DIRREAD authority
  - READDIRREAD, which is the same as granting DIRREAD authority
  - DIRWRITE authority (which implies DIRREAD authority).
- For files within directory control directories:

  Not applicable. Authority on files within directory control directories is derived from the authority on the directory.

2. Granting READ or WRITE authority for a file control directory does not imply any authority to any existing files in that directory. For example, READ authority allows another user to see the name of a file, but not to read that file.
3. When a file control directory is created, or when an existing directory is changed to file control, the owner has WRITE authority and NEWWRITE authority on the directory. Note that file control is the default value for the directory attribute.
4. When a directory control directory is created or when an existing directory is changed to directory control, the owner has DIRWRITE authority.
5. To upgrade READ, NEWREAD, or DIRREAD authority, reissue the DMSGRANT routine with the appropriate option (WRITE, NEWWRITE, or DIRWRITE).
6. To revoke an authority or to downgrade an authority (from WRITE to READ, for instance), use the DMSREVOK routine.
7. When a user grants authority on an alias, the authority refers to the base file.
8. Only the file pool administrator or the owner of the directory or the base file can grant authorities on files or directories.
9. An owner can grant authority to another user for a file even if the file is open, or locked, unless the lock is EXCLUSIVE.
10. When you grant authority on a directory, that authority does not propagate to subdirectories of that directory.
11. Accessing a directory control directory provides distinctive operational characteristics and potential performance improvements. See “DMSCRDIR - Create Directory” on page 88 for more about the characteristics of directory control directories.
12. Granting of authority at a different level than currently exists for a user (upgrading from DIRREAD to DIRWRITE) does not affect the current user access to the directory.
13. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for file or directory on which you are granting authority.
14. If the COMMIT parameter is specified and the return code is 8, then either:
    - An error occurred during the processing of the Grant Authority operation, or
    - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
15. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.

16. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
17. You cannot grant authority to an external object. Anyone who can read the directory can query the remote name in an external object.

Return Codes and Reason Codes

For lists of the possible return codes from DMSGRANT, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSGRANT. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warning cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

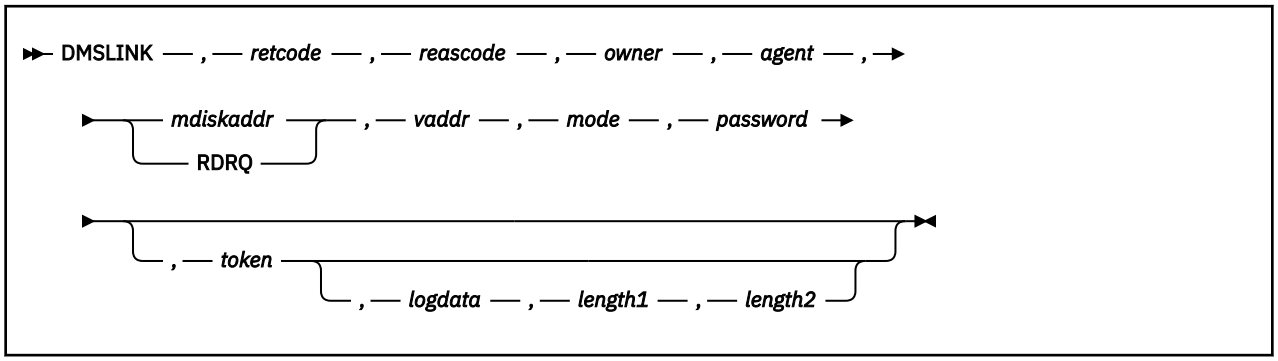
DMSGRANT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	32650	READ authority granted to a user ID that already has WRITE authority.
WARNING	32660	NEWREAD authority granted to a user ID that already has NEWWRITE authority.
WARNING	32670	DIRREAD authority granted to a user ID that already has DIRWRITE authority.
WARNING	32690	The default for the DMSGRANT routine (READ or DIRREAD) caused DIRREAD authority to be granted on a directory control directory.
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	44000	File does not exist or you are not authorized to grant authority to it.
ERROR	44300	Grant failed because object is protected by external security manager.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.

Severity	Reason Code	Description
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	63100	DIRREAD or DIRWRITE authority cannot be specified on a file control directory.
ERROR	63200	DIRREAD or DIRWRITE or NEWREAD or NEWWRITE or READDIRREAD authority parameter cannot be specified for a file.
ERROR	63300	READ, WRITE, NEWREAD, or NEWWRITE was specified for a directory control directory or a file in a directory control directory.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT parameter is specified.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90230	Directory does not exist or you are not authorized to grant authority to it.
ERROR	90310	Invalid option in CSL parameter list. For example, a specified user ID is longer than 8 characters, begins with a plus (+) or a minus (–), or contains a colon (:) or a period (.).
ERROR	90315	Missing option in CSL parameter list.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least one and not more than three tokens in the string.
ERROR	90410	Invalid parameter length specified.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Severity	Reason Code	Description
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSLINK - Link to User Minidisk or Virtual Reader Queue



Call Format

The format for calling a CSL routine is language dependent. DMSLINK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSLINK
(input, CHAR, 8) can be passed as a literal or in a variable. DMSLINK must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSLINK routine in a server virtual machine to obtain and/or verify access to a user's minidisk or virtual reader on behalf of a client.

For minidisks, DMSLINK replaces use of the CP LINK command for authorized servers. It differs from the CP LINK command in that:

1. It does not increment the invalid minidisk password count.
2. The appropriate link mode (RR, MR, and so on) can be determined automatically based on minidisk passwords or an external security manager.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSLINK.

reascode
(output, INT, 4) is a variable which, if *retcode* is 20, contains the CP LINK command error message number.

owner
(input, CHAR, 8) is a variable for specifying the VM user ID of the resource owner.

agent
(input, CHAR, 8) is a variable for specifying the VM user ID on whose behalf access is being requested. If *agent* and *owner* are not the same, then *agent* must have LOGON BY privileges for *owner*, or *password* may be required (see below). If the *agent* is specified as nulls, the *agent* and *owner* are assumed to be the same.

RDRQ
(input, CHAR, 4) indicates that the *agent* wishes to access *owner's* virtual reader queue.

mdiskaddr
(input, CHAR, 4) is a variable for specifying the virtual address of the minidisk as defined in the CP directory entry for *owner*. *mdiskaddr* must be padded on the left with character '0'.

vaddr
(input, CHAR, 4) is a variable for specifying the virtual address of the resource. For minidisks, the device will be linked as the specified virtual address. For virtual reader queues, *vaddr* identifies the address of a defined virtual punch or printer device. *vaddr* must be padded on the left with character '0'.

mode
(input, CHAR, 2) is a variable for specifying the level of access being requested. For minidisks, valid values are the same as for the CP LINK command (RR, MR, and so on). Consult the *z/VM: CP Commands and Utilities Reference* or online help for a complete list of valid link modes. If "X" is specified, you will be given access according to the highest level of access you are permitted, as determined by the resource password, if any, or by an external security manager. The level of access given may be affected by the number and type of links that already exist. For virtual reader queues, *mode* is ignored and is treated as though "X" was specified. If a single character is specified, it must be padded on the right with a blank.

password
(input, CHAR, 8) is a variable for specifying the resource password. This field may be specified as nulls when *agent* and *owner* are the same or if public access is permitted. If native CP security services are used, the password applies only to minidisks.

token
(input, CHAR, 4) is a variable for specifying a security token. If *token* is omitted, or the value of the token is zero, then native CP security services will be used to validate the user information. If the value of the token is nonzero, standard external security manager services will be used. If the value of the token is 1 (0x00000001) then external security services will be used, but the token will not be passed to the security service.

logdata
(output, CHAR, *length1*) is a variable which may contain arbitrary readable text passed from the CSL routine to the calling application. This text is used to more fully describe any error conditions.

Note: This text may or may not be used by the calling application.

length1
(input, INT, 4) is a variable for specifying the length of the *logdata* field. A value of zero indicates that no text is to be returned by DMSLINK. The maximum value is 256.

length2
(output, INT, 4) is a variable in which DMSLINK specifies the length of the text that it placed in *logdata*. A value of zero indicates that no text is present.

Usage Notes

1. The issuing virtual machine must have `OPTION DIAG88` and `privilege class B` specified in the CP directory entry.
2. DMSLINK communicates with the external security manager using the RACROUTE macro interface, as shown in the following table:

Resource type	RACROUTE macro parameters	Minimum Access Required
Virtual reader	REQUEST=AUTH USERID=*agent* CLASS=VMRDR ENTITYX=*owner or acigroup.owner*	UPDATE

Resource type	RACROUTE macro parameters	Minimum Access Required
Minidisk	REQUEST=AUTH USERID=*agent* CLASS=VMMDISK ENTITYX=*owner.mdiskaddr* or *acigroup.owner.mdiskaddr*	READ

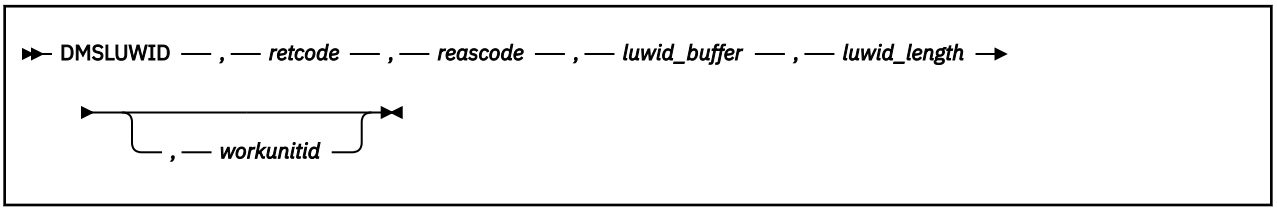
Information about RACROUTE can be found in the *z/VM: Security Server RACROUTE Macro Reference*.

Return Codes and Reason Codes

The following table lists the DMSLINK return codes.

Return Code	Description
0	Full access is permitted: • For a minidisk, a R/W link was established. • For a virtual reader queue, *agent* may view, delete from, or add to the queue.
4	Limited access is permitted: • For a minidisk, a R/O link was established. • For a virtual reader queue, *agent* may only add to the queue.
8	No access allowed.
12	Resource password is required, but was not supplied.
16	Resource password is incorrect.
20	System error – For minidisks, reason code contains error message number from the CP LINK command.
24	System paging I/O error.
32	Function not authorized. See Usage Note "1" on page 292.
36	Function not available.
-1nn	Parameter *nn* is not valid.

DMSLUWID - Get a Logical Unit of Work ID



Context

Coordinated Resource Recovery (CRR)

Call Format

The format for calling a CSL routine is language dependent. DMSLUWID is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSLUWID
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSLUWID routine to obtain the logical unit of work ID (LUWID) associated with a work unit.

DMSLUWID is meaningful only for protected conversations. Applications must make sure that this function is only invoked for a work unit having protected conversations.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSLUWID. For lists of the possible return codes, see Appendix D, "Return Codes," on page 597.

reascode
(output, INT, 4) is a variable for the reason code from DMSLUWID. For a list of common CSL reason codes that can occur with most file system management and related routines, see "Common Reason Codes" on page 601.

LUWID_buffer
(output, CHAR, 26) is a variable used to return the current LUWID associated with the specified (or default) CMS work unit ID.

If no LUWID is associated with the specified (or default) CMS work unit ID, one will be created and associated with it.

LUWID_length
(output, INT, 1) is a variable used to return the actual length of the LUWID.

workunitid
(input, INT, 4) is a variable for specifying the CMS work unit ID for which the associated LUWID is being requested. If you do not specify *workunitid*, DMSLUWID returns the LUWID associated with the current default CMS work unit.

Usage Notes

1. The Logical Unit of Work Identifier can be up to 26 bytes long, containing the following fields:

   Length
   Description

   1
   length of the fully-qualified LU name

   1-17
   the fully-qualified LU name. If its length is less than 17 bytes, it is left-justified and padded on the right with blanks (X'40'). It is composed of the following fields:

   Length
   Description

   0-8
   the SNA network ID

   0-1
   a delimiter (a period)

   1-8
   the LU name

   If both the network ID and the LU name are present, they are separated by a period.

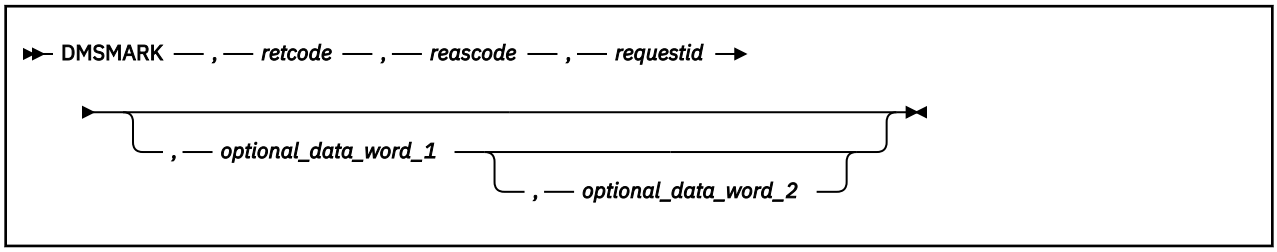
   6
   Instance number in binary

   2
   Sequence number in binary

   For more information about the logical unit of work identifier, see the *SNA Format and Protocol Reference Manual for LU Type 6.2*, SC30-3269.

2. DMSLUWID can be useful if an application needs to monitor its own resources, for example, for accounting purposes. This routine can also be used by an application designed to act as its own protected conversation adapter to get an LUWID in order to register conversations.

DMSMARK - CRR Mark Request ID



Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSMARK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSMARK
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSMARK" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSMARK routine to mark the completion of an asynchronous event. CRR provides this capability for multitasking applications, or for applications doing parallel work in several resources. The SPM creates a request ID for the resource during registration and passes the request ID to the resource adapter's exit each time the exit is taken. If the resource adapter must wait for some asynchronous event to complete, the resource adapter's exit routine can return the ADAREDRV (Redrive) return code to the SPM. When the asynchronous event completes, the resource adapter calls DMSMARK to signal the completion of the event to the SPM.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSMARK.

reascode
(output, INT, 4) is a variable for the reason code from DMSMARK.

requestid
(input, INT, 4) is a variable that identifies the asynchronous request (event) to be marked as complete.

optional_data_word_1
(input, INT, 4) is a variable that contains general purpose data to be passed to the creator of the request ID. For SPM request IDs, this word is the resource adapter's response to an SPM broadcasted action. If a value of 0 is specified, it is the same as no response.

optional_data_word_2
(input, INT, 4) is a variable that contains additional general purpose data to be passed to the creator of the request ID. For SPM request IDs, this word is used to update the actual error data length to reflect additional error or warning information in the resource adapter's error block. If a value of 0 is specified, no changes are made.

Usage Notes

1. For guidance information on using the DMSMARK routine in the context of getting a resource manager to participate in CRR, see *z/VM: CMS Application Development Guide*.
2. The application should not use DMSMARK. It is intended for use by CRR resource adapters and by the system. The system uses this routine to make sure the CPI Communications XCWOE (Wait On Event) routine detects and returns SPM asynchronous events.
3. For SPM request IDs, an optional data word value of 0 has the same effect as if the optional data word were omitted. For example, the resource adapter could pass a new actual error data length in optional data word 2, but pass a 0 (no response) in optional data word 1. The SPM would update the actual error data length for the resource adapter and still redrive the resource adapter's exit. This scenario is not intended to be typical but is permitted.
4. If a nonzero response is given in optional data word 1, this means the resource adapter's exit should not be redriven. If no resource adapter response is given, the caller's resource adapter exit will be redriven by the SPM at a later time. For more information, see *z/VM: CMS File Pool Planning, Administration, and Operation*.
5. The error block and actual error data length were passed to the resource adapter when its sync point exit was driven. If there are new errors or warnings, they should be appended to the error block, and the new actual error data length should be passed in optional data word 2.
6. No error is reported if more optional data words are passed than the request ID is capable of accepting. For example, if a request ID is not expecting optional data words, any that are passed to DMSMARK are ignored.
7. Resource adapters should use the CSL *fast path* to call DMSMARK. For more information about the CSL fast path, which uses the CSLFPI macro, see the *z/VM: CMS Macros and Functions Reference*.

Return Codes and Reason Codes

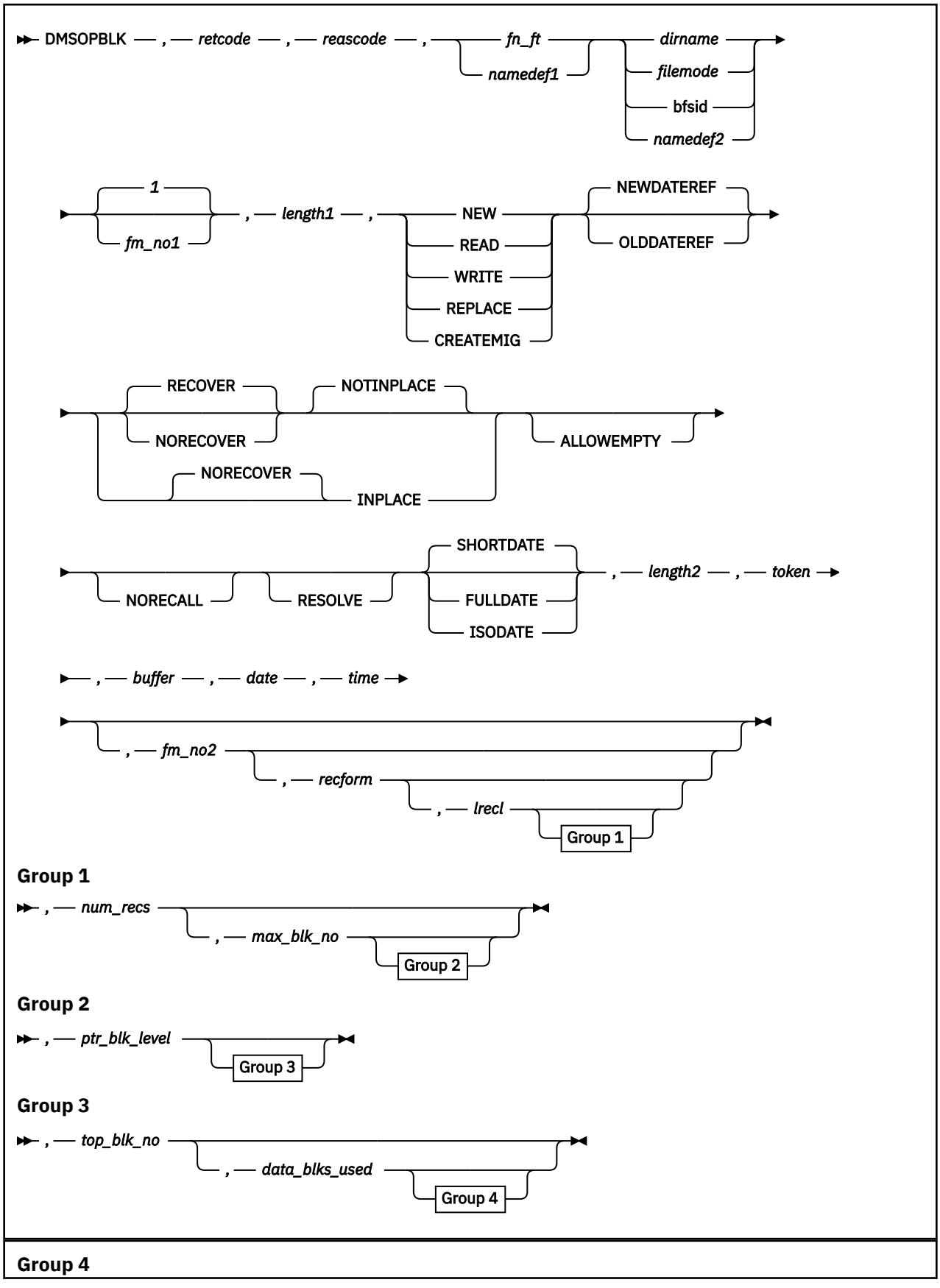
For lists of the possible return codes from DMSMARK, see Appendix D, “Return Codes,” on page 597.

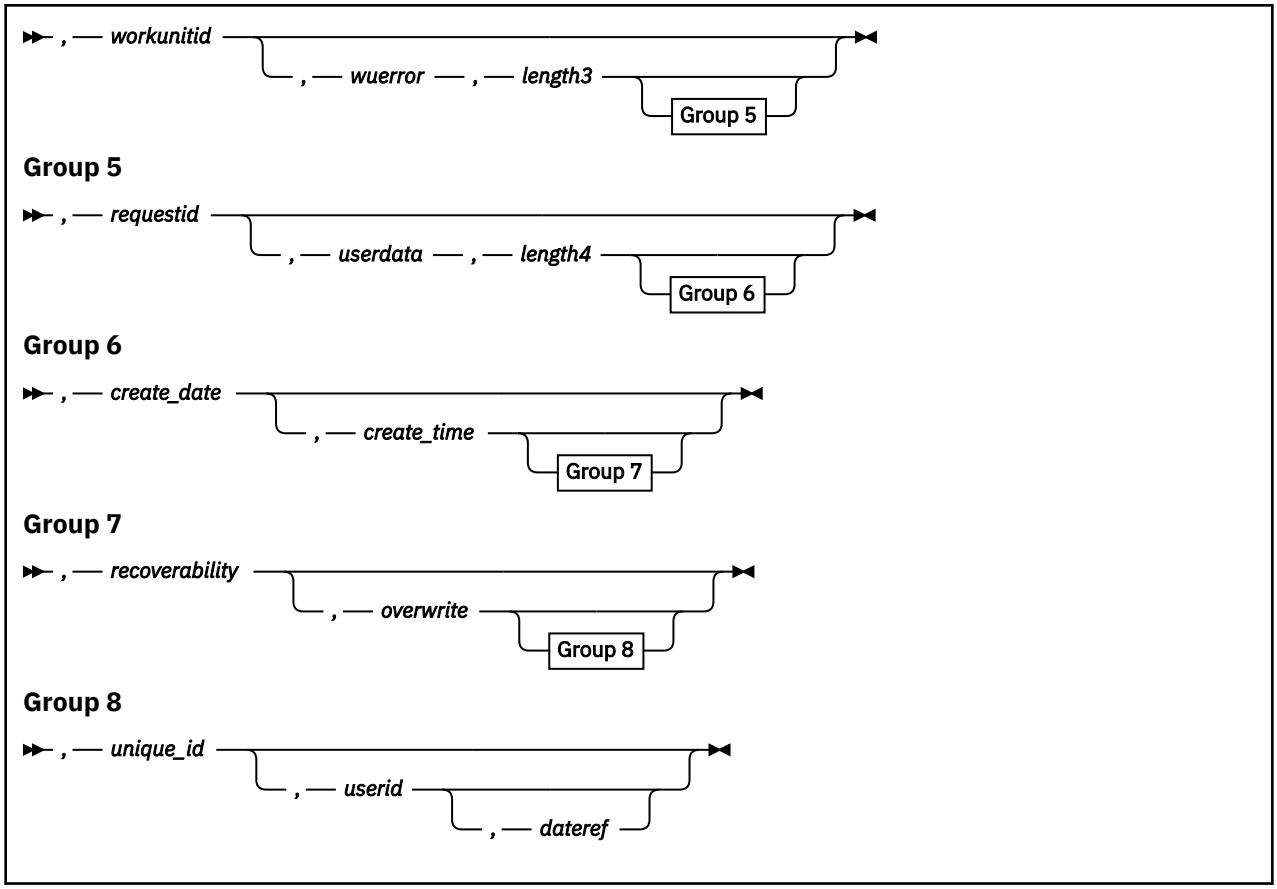
The following table lists the special reason codes returned by DMSMARK. WARNING means the request was processed, but a warning condition was encountered; ERROR means the request failed. Warnings cause a return code of 4, and errors cause a return code of 8.

DMSMARK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	90570	Request ID has already been marked. Any optional data words have not been modified by DMSMARK.
ERROR	90216	Invalid *requestid* parameter. There is no such request ID active at this time.

DMSOPBLK - Open Blocks





Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSOPBLK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSOPBLK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSOPBLK routine to prepare an SFS or BFS file for subsequent use by Read Blocks (DMSRDBLK), Write Blocks (DMSWRBLK), or Close Blocks (DMSCLBLK).

The block routines are intended for backup and restoration operations. Their use requires extensive knowledge of and experience with CMS file structure.

The structure of a CMS file stored in SFS is nearly identical to the structure used for Enhanced Disk Format (EDF) files. The disk addresses in the pointer blocks are not, however, relative to the beginning of a minidisk, as they are for EDF files. Instead, they are relative to the beginning of the file. If you are not knowledgeable about CMS file structure, using the block routines may result in unusable files. For routine file I/O, use DMSOPEN, DMSREAD, DMSWRITE, and DMSCLOSE.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSOPBLK.

reascode
(output, INT, 4) is a variable for the reason code from DMSOPBLK.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file being opened. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the SFS directory where the file to be opened exists (or will exist).

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed SFS directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode. If the file mode letter is not an accessed SFS directory, an error return code will result.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system containing the file to be opened.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

fm_no1
(input, CHAR, 1) is a variable for specifying the file mode number assigned when a file is opened with an intent of NEW, REPLACE, WRITE (for a new file), or CREATEMIG. The default is 1 when a new file is created. If this parameter is omitted for REPLACE or CREATEMIG of an existing file, the previous file mode number of the file is used. This parameter is ignored when an existing file is opened with an intent of READ or WRITE.

If a file mode number specified for REPLACE, and it is different from the existing file mode number, then the file mode number of the base file and all aliases of that base file are changed to the specified value.

For a BFS file, the only file mode number allowed is 1.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*; plus *dirname*, *filemode*, *bfsid*, or *namedef2*; plus *fm_no1*, if specified). See “Compound Variables” on page 15 for coding details.

NEW
(input, CHAR, 3) means to open a new file (one that does not already exist). The file is created and can now be written to and read from. This intent is not allowed for a BFS file.

READ
(input, CHAR, 4) means that the file can only be read and it must already exist. It is an error to Open Blocks for READ for a file that does not exist.

REPLACE
(input, CHAR, 7) means that if the file exists, it will be replaced with only the added blocks. If a specified SFS file does not exist, a new file is created. This intent is not allowed for a BFS file if the file does not already exist.

WRITE
(input, CHAR, 5) means the file may be written to or read from. All changed and added blocks will be written. Other records remain unchanged. If the file does not exist, it is created. This intent is not allowed for a BFS file.

CREATEMIG
(input, CHAR, 9) creates an empty migrated file in preparation for restoring it. If the file exists, it is replaced. Only Close Blocks (DMSCLBLK) can be issued against the file; it cannot be read from or written to. File pool administration authority is required to use CREATEMIG.

This intent is not allowed for a BFS file if the file does not already exist.

NEWDATEREF
(input, CHAR, 10) causes SFS to update the date of last reference for the file for any open intent. The date of last reference is the date the file was last read or modified. NEWDATEREF is the default.

OLDDATEREF
(input, CHAR, 10) causes SFS not to change the date of last reference when the file is opened with an intent of READ. The date of last reference is the date the file was last read or modified. OLDDATEREF is ignored when the file is opened with intents other than READ, because they allow the file to be changed.

RECOVER
(input, CHAR, 7) specifies that all uncommitted changes are to be discarded when a work unit is rolled back. The rollback can be initiated by the application, or it can be caused by abend processing or system failure.

The recoverability attribute (RECOVER or NORECOVER) is assigned to the file only when the open intent is NEW, REPLACE, WRITE (for a nonexistent file), or CREATEMIG. If the recoverability attribute is not specified for one of these intents, a value is assigned. See usage notes "18" on page 307 and "20" on page 307. If an overwrite attribute of INPLACE is specified or assigned, the default recoverability attribute is NORECOVER.

When an alias is opened for REPLACE, the recoverability of the base file and all aliases on the base is set to RECOVER.

NORECOVER
(input, CHAR, 9) specifies that changes to the file are not to be rolled back when the application initiates a rollback. In most cases, the updates are committed.

In implicit rollbacks, changes are committed if possible; in particular, cached updates can be discarded.

The recoverability attribute (RECOVER or NORECOVER) is assigned to the file only when the open intent is NEW, REPLACE, WRITE (for a nonexistent file), or CREATEMIG. If the recoverability attribute is not specified for one of these intents, a value is assigned. See usage notes "18" on page 307 and "20" on page 307. If an overwrite attribute of INPLACE is specified or assigned, the default recoverability attribute is NORECOVER.

When an alias is opened for REPLACE, the base file and all aliases on the base are set to NORECOVER.

This intent is not allowed for a BFS file.

NOTINPLACE
(input, CHAR, 10) specifies that readers see a consistent version of the file from Open to Close.

The overwrite attribute (NOTINPLACE or INPLACE) is assigned to the file only when the open intent is NEW, REPLACE, WRITE (for a nonexistent file), or CREATEMIG. If the overwrite attribute is not specified for one of these intents, a value is assigned. See usage notes "18" on page 307 and "20" on page 307.

When an alias is opened for REPLACE, the base file and all aliases on the base are set to NOTINPLACE.

INPLACE
(input, CHAR, 7) specifies that updates are to be made in place where possible. This can reduce DASD utilization, and enables readers to see file updates by a concurrent writer. Users that have an INPLACE file open for reading have to reopen the file to see extensions (new records or blocks) that have been written and committed to the file.

The overwrite attribute (NOTINPLACE or INPLACE) is assigned to the file only when the open intent is NEW, REPLACE, WRITE (for a nonexistent file), or CREATEMIG. If the overwrite attribute is not specified for one of these intents, a value is assigned. See usage notes "18" on page 307 and "20" on page 307.

When an alias is opened for REPLACE, the base file and all aliases on the base are set to INPLACE.

This intent is not allowed for a BFS file.

ALLOWEMPTY
(input, CHAR, 10) permits a file with no records to be created by this DMSOPBLK request. If there are no blocks in the file when it is closed, either a new empty file is created (SFS only), or an existing file is replaced with an empty file. ALLOWEMPTY is implied with CREATEMIG and need not be specified.

NORECALL
(input, CHAR, 8) causes the request to fail if the file has been migrated to the DFSMS/VM repository. The file is not recalled. NORECALL applies only for READ and WRITE; it is ignored for NEW, REPLACE, and CREATEMIG. If this option is omitted, the file can be recalled if SET RECALL has not been set to OFF.

RESOLVE
(input, CHAR, 7) causes an unresolved alias to be resolved when work is committed for the file (see usage note "24" on page 307). RESOLVE applies to NEW, REPLACE, WRITE, and CREATEMIG intents; it is ignored for READ. RESOLVE is implied for CREATEMIG and need not be specified.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date*, *create_date*, and *dateref* parameters is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date*, *create_date*, and *dateref* parameters is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date*, *create_date*, and *dateref* parameters is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input; INT; 4) specifies the length of the preceding compound character parameter (NEW, READ, WRITE, REPLACE, or CREATEMIG; NEWDATEREF or OLDDATEREF, if specified; RECOVER or NORECOVER, if specified; NOTINPLACE or INPLACE, if specified; ALLOWEMPTY, NORECALL, and RESOLVE, if specified; SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

token
(output, CHAR, 8) is returned to your program and identifies this particular open request. You pass the token on requests to Read Blocks (DMSRDBLK), Write Blocks (DMSWRBLK), and Close Blocks (DMSCLBLK) to relate them to this open request.

buffer
(output, CHAR, 4KB) is a 4K-byte area into which the first (top) block of the file is placed if an existing file is opened for READ or for WRITE. Note that the block returned may be a pointer block. If the file is empty or opened new, the buffer is not changed.

date
(output, CHAR, 8 or 10) is the date on which the file was last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month. If the file is new, DMSOPBLK sets the date to blanks.

You must specify either the FULLDATE or the ISODATE parameter if you want 4-digit years returned. SHORTDATE is the default, which is the 2-digit year format.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

time
(output, CHAR, 8) is the time at which the file was last updated. The format is *hh:mm:ss*. If the file is new, DMSOPBLK sets the time to blanks.

fm_no2
(output, CHAR, 1) is the file mode number returned by DMSOPBLK.

recform
(output, CHAR, 1) indicates whether the file consists of fixed-length or variable-length records. The following values can be returned:

F
indicates that all the records in the file have the same length.

V
indicates that the records in the file may have different lengths.

lrecl
(input/output, INT, 4) is the logical record length. For a BFS file, the only value allowed is 1.

num_recs
(output, INT, 4) is the number of records in the file.

max_blk_no
(output, INT, 4) is the largest block number for the file.

ptr_blk_level
(output, INT, 4) is the number of pointer block levels used.

top_blk_no
(output, INT, 4) is the block number of the first block of the file.

data_blks_used
(output, INT, 4) is the number of data blocks used in the file.

workunitid
(input, INT, 4) specifies the work unit ID to be associated with this routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

userdata
(input, CHAR, 1-80) is a variable specifying a string of user data to be passed to an external security manager (ESM). The format and meaning of the data is defined by the ESM (see the Usage Notes).

length4
(input, INT, 4) is a variable containing the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

create_date
(input/output, CHAR, 8 or 10) is a variable that serves as either an input or an output depending on whether the file already exists. It is based on Coordinated Universal Time (UTC).

If the file exists, *create_date* returns the date on which the file was created.

You cannot set the creation date for existing files. If you specify the creation date when you open an existing file for WRITE or REPLACE, the actual creation date is returned, and a warning reason code is issued. To avoid the warning, set *create_date* to blanks.

Note that you must specify 10-character output fields if you specify FULLDATE or ISODATE; otherwise, you could get storage overlays.

If the file does not exist (you are creating it), you can use this parameter to set the date of file creation. You can have the system determine the creation date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *create_date* variable, or leave blanks, the system generates the creation date.

The *create_date* is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years (input), or have 4-digit years returned (output).

When specifying *create_date*, use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

For existing files, the *creation_date* is returned to you in the same form.

DMSOPBLK checks the format of the date, but does not check that the date itself is valid or reasonable.

create_time
(input/output, CHAR, 8) is a variable that serves as either an input or an output, depending on whether the file already exists. It is based on Coordinated Universal Time (UTC).

If the file exists, *create_time* returns the time at which the file was created.

You cannot set the creation time for existing files. If you specify the creation time when you open an existing file for WRITE or REPLACE, the actual creation time is returned, and a warning reason code is issued. To avoid the warning, set *create_time* to blanks.

If the file does not exist (you are creating it), you can use this parameter to set the time of file creation. You can have the system determine the creation time by omitting the parameter or by specifying 8 blanks.

Specify the time in the character format *hh:mm:ss*. For existing files, the creation time is returned to you in the same form. Use a colon (:) as the separator character and 2 digits in each position. For example:

```
15:12:20
```

DMSOPBLK checks the format of the time, but does not check that the time itself is valid.

recoverability
(output, CHAR, 1) is a variable in which a value is returned that indicates whether the recoverability attribute is RECOVER or NORECOVER after DMSOPBLK processing:

1
indicates RECOVER

0
indicates NORECOVER

Usage note "18" on page 307 describes how the value of the recoverability parameter is determined when a new file is opened with NEW, WRITE, or CREATEMIG.

overwrite
(output, CHAR, 1) is a variable in which a value is returned that indicates whether the overwrite attribute is NOTINPLACE or INPLACE after DMSOPBLK processing:

1
indicates INPLACE

0
indicates NOTINPLACE

Usage note "18" on page 307 describes how the value of the overwrite parameter is determined when a new file is opened with NEW, WRITE, or CREATEMIG.

unique_id
(output, CHAR, 24) is the unique identifier for the primary repository file. The content of *unique_id* is a 16 character unique ID and a 1 to 8 character file pool ID. It is meaningful only when the intent is CREATEMIG; blanks are returned for other intents.

userid
(input, CHAR, 8) is a user ID that the file pool server uses to bypass explicit exclusive locks on a storage group or file space when a file is opened with the CREATEMIG intent. DMSOPBLK can create the empty migrated file despite any explicit exclusive locks held by *userid*. The parameter is ignored with the other open intents. Specifying blanks has the effect of omitting the *userid* parameter.

dateref
(input, CHAR, 8 or 10) is a variable for specifying the date of last reference. The date of last reference is the date the file was created, last read, or last updated.

If the file does not exist (you are creating it), you can use this parameter to set the date of last reference. You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. The system will generate the date in Universal Coordinated Time (UTC).

It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

DMSOPBLK checks the format of the date, but does not check that the date itself is valid or reasonable.

Usage Notes

1. If the intent is NEW, WRITE, REPLACE, or CREATEMIG for a file that does not exist, the following parameters are set to 0 (if integer) or blank (if character). If the file does exist and the intent is WRITE, REPLACE, or READ, these parameters are set to reflect the attributes of the file opened.

   Exception: The *unique_id* parameter is set when the intent is CREATEMIG.

   date
   time
   fm_no2
   *recform**
   *lrecl**
   *num_recs**
   *max_blk_no**
   *ptr_blk_level**
   *top_blk_no**
   *data_blks_used**
   recoverability
   overwrite
   unique_id

   Note: When the intent is NEW, some of these attributes (marked with an * in the list) are set but not returned.

   CMS does not update these parameters after opening the file. You must ensure that these parameters contain the correct values when you pass them to Close Blocks to close the file. The only case in which you do not have to provide these attributes with Close Blocks is when the file was opened for READ.
2. If you have opened files with DMSOPBLK and modified them, you cannot issue a COMMIT request for the work unit until you have closed the files.
3. Files opened and changed by the Open Blocks routine must be explicitly closed through the Close Blocks routine (DMSCLBLK). If such a file is left open when CMS end-of-command processing gains control, a rollback occurs for the work unit on which the Open Blocks was issued. All uncommitted work on all files on that work unit is lost.
4. You will receive an error when attempting to use the Open Blocks routine with the READ parameter on a file that does not exist.
5. If a file is opened for REPLACE, and then closed before any blocks are written, the file remains unchanged.
6. If a file is opened for REPLACE, it is effectively empty. Only records written after the Open Blocks can be read, and before the first Write Blocks request (DMSWRBLK), Read Blocks (DMSRDBLK) gives an end-of-file return code.
7. If a file or its directory is locked, an attempt to open the file for NEW, REPLACE, or WRITE fails unless the lock is UPDATE or EXCLUSIVE owned by the issuer.
8. A file can be opened with the READ intent regardless of locks, except an EXCLUSIVE lock held by another user on the file or its directory.
9. A request ID is returned if the request is to be asynchronous.
10. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
11. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
12. When you read a file in a directory control directory, the level of the file you see varies depending on whether the directory is accessed. If you open a file for read and you have the directory accessed in read-only status, generally the only changes you will see are those that were committed before you

accessed the directory. If you do not have the directory accessed or you release the directory (with the CMS RELEASE command), you see any changes that were committed before you opened the file.

If you have the directory accessed in read/write status, you are the only user who can be changing files within the directory. You can see your own uncommitted changes on the same work unit.

Files with the INPLACE attribute do not follow the above rules. You can see committed and uncommitted updates to INPLACE files without reaccessing the directory or reopening the file.

13. You cannot open a file with the NEW, REPLACE, or WRITE option if:
    - Another user has the file or directory locked
    - The file resides in a directory control directory and another user has the directory accessed in read/write mode.
    - The file resides in a directory control directory, and you have the directory accessed in read-only mode.
14. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file you are opening.
15. If you specify OLDDATEREF or NEWDATEREF and execute the routine against a file pool that does not support dates of last reference, the parameters are ignored.
16. If you specify *create_date* for a file maintained by a file pool server that does not support the parameter, the character string "00/00/00" is returned. If you specify *create_time* and it is not supported, the character string "00:00:00" is returned.

    Zeros are also returned if the file does not yet have the attribute established for it (perhaps the server has only recently added the support for the attribute).
17. The date and time of creation and the date of last reference are recorded in Coordinated Universal Time (UTC). The date and time of last update are the local time of the CMS user machine.
18. If you do not specify recoverability and overwrite attributes for a file that is being created by this DMSOPBLK, then those file attributes will be based on the defaults for files with that file mode number as established by the DMSPUSHA (Push Attributes) CSL routine. If a Push Attributes has not been done, then the recoverability file attribute defaults to RECOVER, and the overwrite attribute defaults to NOTINPLACE.
19. If the overwrite and recoverability specifications on DMSOPBLK READ do not match those of the base file, a warning return code and reason code are issued.
20. If the file mode number, recoverability, or overwrite attribute is specified with DMSOPBLK REPLACE, the replaced file uses the new attribute as specified. Otherwise, the attribute is inherited from the file being replaced, if one exists. The defaults set by Push Attributes do not apply when replacing an existing file.
21. When using DMSOPBLK WRITE to an existing file, recoverability and overwrite specifications are not used to set the extended attributes of the file. However, if they are specified and they do not match those already in the file, a warning return code and reason code are issued.
22. Files that are NORECOVER or INPLACE are not handled as such when opened by DMSOPBLK. Only when the files are closed and committed do the updates become nonrecoverable. All updates to the file resulting from the Open will be treated as RECOVER and NOTINPLACE. Once the file is closed and committed, the changes to the file and any subsequent activity will be handled in a manner consistent with the values assigned to the recoverability and overwrite attributes.
23. When *wuerror* has been specified on DMSOPBLK and an SFS error occurs during the DFSMS/VM file recall process, FPERROR is updated to include the specific reason code of the failing request and the file pool ID of the failing resource.
24. DMSCRALI can create an "unresolved alias" when the base file does not exist and when the user lacks the necessary authorizations to the base file and target directory (see usage note “4” on page 83). If RESOLVE is specified on the DMSOPBLK request, when the work unit is committed:

- Any unresolved alias for which proper authorizations exist is converted to an alias
- Any unresolved alias for which proper authorizations do not exist is converted to a revoked alias, and a warning reason code is returned.

25. No attempt is made to resolve aliases when the work unit is committed before anything has been written:
    - To an existing file that was opened for WRITE
    - Or to new file that was created without the ALLOWEMPTY option.
26. If CREATEMIG is specified for a file in a file pool that does not support creating migrated files, an error reason code is returned.
27. When the DMSOPBLK CREATEMIG request refers to a file that is not in the primary repository, the file is created in the primary repository as a migrated file with the file attributes specified on the DMSCLBLK request.
28. BFS files may contain more than 2^{31}-1 records (bytes) when created by means other than CMS record interfaces. When opening a BFS file with CMS record interfaces, the open attempt will fail if the file has more than 2^{31}-1 records. An attempt to write more than 2^{31}-1 records will also fail.
29. Opening a BFS file using this interface requires file pool administration authority.
30. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date*, *create_date*, and *dateref* parameters.
31. If DMSOPBLK is called from a program written in REXX, the *date* and *create_date* fields returned as output will always be 10 characters in length. If the SHORTDATE format is specified or allowed to default, these fields will be padded on the right with 2 blanks.
32. When *create_date* (given as input) and *dateref* are specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
33. When *create_date* (given as input) and *dateref* are specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
34. If you want to perform arithmetic or conversion operations on the time stamps that are input or output by this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSOPBLK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSOPBLK. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSOPBLK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	10220	File mode value other than 1 specified for a BFS file.

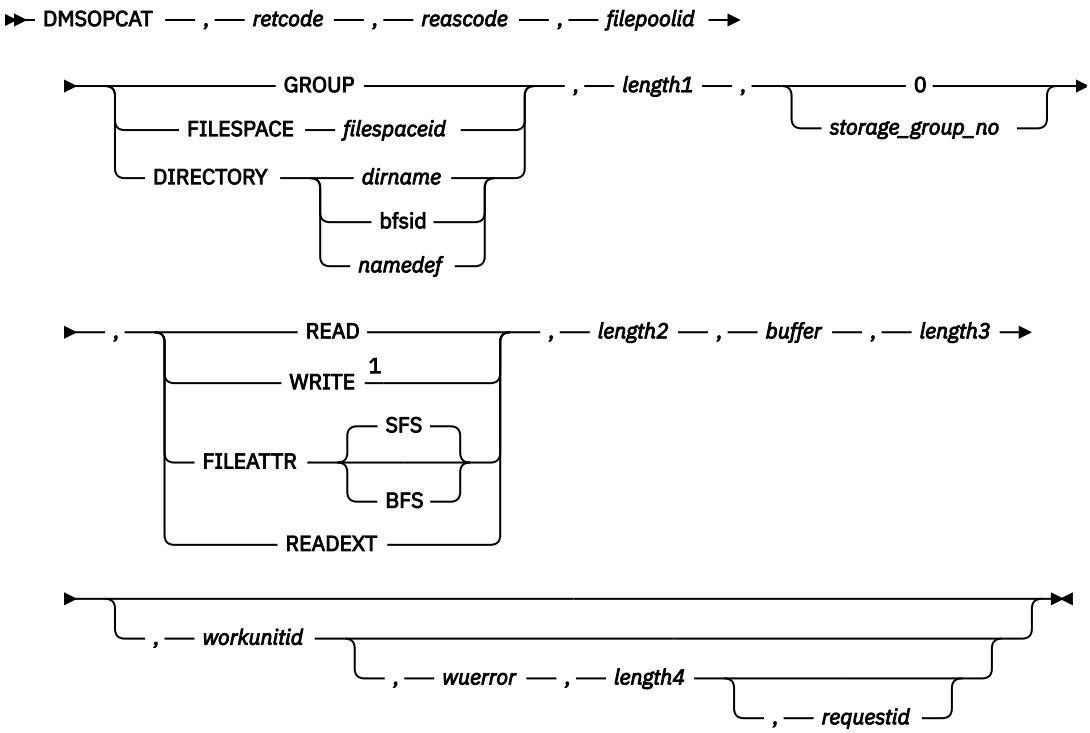
Severity	Reason Code	Description
WARNING	44030	Intent was WRITE or REPLACE, and the file did not previously exist.
WARNING	51051	You specified a nonblank value for *create_date* or *create_time* and opened an existing file with an intent of REPLACE or WRITE. You cannot change the creation date or time of an existing file.
WARNING	78107	Recoverability attribute does not match that of the base file. For a BFS file, the NORECOVER attribute is not allowed.
WARNING	78108	Overwrite attribute does not match that of the base file. For a BFS file, the INPLACE attribute is not allowed.
WARNING	98700	Server managing this file does not support certain parameters of DMSOPBLK: **NORECOVER** RECOVER is assumed. **INPLACE** NOTINPLACE is assumed. **ALLOWEMPTY** The server ignores the parameter. **RESOLVE** The server ignores the parameter. ***create_date*** The server ignores the parameter. ***create_time*** The server ignores the parameter. ***dateref*** The server ignores the parameter.
ERROR	20000	Intent was NEW, but the file already exists.
ERROR	30000	You do not have the required authority. File pool administration authority is required to use the CREATEMIG intent or to open a BFS file.
ERROR	30500	The combination of INPLACE and RECOVER is not supported.
ERROR	31000	Mixing recoverable and nonrecoverable updates for the same file is incorrect within a single work unit.

Severity	Reason Code	Description
ERROR	44000	• Intent was NEW or CREATEMIG, and the directory does not exist. • Intent was NEW, and you are not authorized to create a file in the directory. • Intent was WRITE or REPLACE, and the file does not exist and you are not authorized to create a file in the directory. • Intent was WRITE or REPLACE, and you are not authorized to write to the file. • Intent was WRITE or REPLACE, and the directory does not exist. • Intent was READ, and the file does not exist or you are not authorized to read the file.
ERROR	44200	Your attempt to open an SFS file with an intent of NEW, WRITE, REPLACE, or CREATEMIG failed because you already have the file open with one of these intents on a DMSOPEN, DMSOPBLK, or DMSOPDBK request. This reason code is returned only if both attempts to open the file occurred on the same work unit.
ERROR	44700	Input file has been move and there is no active DFSMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)
ERROR	50500	Attempt to exceed the maximum number of 4KB file space blocks allowed for this user. Applicable only when NEW, WRITE, REPLACE, or CREATEMIG parameter is specified.
ERROR	50600	File cannot be opened. The user has no storage blocks allocated to his file space.
ERROR	63700	Directory control directory is accessed read-only.
ERROR	63800	The specified directory is a directory control directory. The CREATEMIG keyword is not supported for directory control directories.
ERROR	65200	File is in DFSMS/VM migrated status and implicit RECALL is set to OFF. (Set RECALL ON or issue a DFSMS RECALL command).
ERROR	65400	Request cannot be performed on a BFS object because of one or more of the following: • Intent was not READ or REPLACE for a BFS file. • BFS file has more than 2^{31}-1 records (bytes). • Intent was REPLACE for a BFS file and the file does not exist.
ERROR	65500	File is in DFSMS/VM migrated status and there is no active DFSMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)
ERROR	66100	File is in DFSMS/VM migrated status and DFSMS/VM recall processing had been disabled.
ERROR	90210	Extraneous characters present in an input parameter.
ERROR	90250	The file name and file type (or *namedef*) are required but were not specified.

Severity	Reason Code	Description
ERROR	90310	Incorrect option in CSL parameter list. Valid options are: NEW, READ, WRITE, REPLACE or CREATEMIG; NEWDATEREF or OLDDATEREF; RECOVER or NORECOVER; INPLACE or NOTINPLACE; ALLOWEMPTY; NORECALL; RESOLVE; and SHORTDATE, FULLDATE, or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. NEW, READ, WRITE, REPLACE, and CREATEMIG are mutually exclusive parameters, NEWDATEREF and OLDDATEREF, if specified, are mutually exclusive parameters, RECOVER and NORECOVER, if specified, are mutually exclusive parameters, INPLACE and NOTINPLACE, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: NEW, READ, WRITE, REPLACE, CREATEMIG, NEWDATEREF, OLDDATEREF, RECOVER, NORECOVER, INPLACE, NOTINPLACE, ALLOWEMPTY, NORECALL, RESOLVE, SHORTDATE, FULLDATE, or ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the file ID or directory name parameter. There must be at least 1 and not more than 4 tokens in the string.
ERROR	90380	Missing parameter in CSL parameter list.
ERROR	90410	Incorrect length specified for one of the character variables.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the file ID parameter is incorrect. The file name is longer than eight characters or contains an incorrect character.
ERROR	90430	The file type in the file ID parameter is incorrect. The file type is longer than eight characters or contains an incorrect character.
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Incorrect characters (* or %) were found in either the file name or file type part of the file ID parameter.
ERROR	90472	Incorrect request ID specified, must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	Nonnumeric value in date specification.
ERROR	90498	Incorrect time format; must be in the form *hh:mm:ss*.
ERROR	90499	Nonnumeric value in time specification.

Severity	Reason Code	Description
ERROR	90500	The specified directory name is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the file ID or directory name parameter is longer than 16 characters.
ERROR	90530	The namedef part of the file ID or directory name parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name or file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and file pool ID was not specified as part of the directory name.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	90604	You have already opened the SFS file with the FS macro interface for output.
ERROR	98700	CREATEMIG keyword is not supported by this file pool.

DMSOPCAT - Open Catalog



Notes:

[1] You cannot open the catalog of a directory for writing.

Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSOPCAT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSOPCAT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSOPCAT routine to open a user storage group or file space to read or write the catalog information, or open a directory to read the catalog information. File pool administration authority is required to specify a storage group. For an SFS file space or directory, ownership or file pool administration authority is required. For a BFS file space, file pool administration authority is required.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable to hold the return code from DMSOPCAT.

reascode
(output, INT, 4) is a variable to hold the reason code from DMSOPCAT.

filepoolid
(input, CHAR, 1-8) is a variable for specifying the name of the file pool containing the object to be opened. When specifying this parameter, an appended colon is not valid and should not be used.

GROUP
(input, CHAR, 5) means that the catalog for the storage group is to be opened.

FILESPACE
(input, CHAR, 9) means that the catalog for the file space is to be opened.

filespaceid
(input, CHAR, 1-8) is a variable for specifying the name of the affected file space. This parameter can be used only when FILESPACE is specified. When specifying this parameter, note that nicknames are not allowed and should not be used.

DIRECTORY
(input, CHAR, 9) means that the catalog for the directory is to be opened.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the affected SFS directory. This parameter can be used only when DIRECTORY is specified.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system (BFS top directory). This parameter can be used only when DIRECTORY is specified.

namedef
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname* or *bfsid*. This parameter can be used only when DIRECTORY is specified.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*filepoolid*; plus GROUP, FILESPACE *filespaceid*, DIRECTORY *dirname*, DIRECTORY *bfsid*, or DIRECTORY *namedef*). See "Compound Variables" on page 15 for coding details.

storage_group_no
(input, INT, 4) is for specifying the storage group affected. It is a value between 2 and 32767, and not greater than the MAXDISKS parameter used on the FILESERV GENERATE command for the file pool. This is specified when the object is GROUP. If the catalog is not opened for a storage group, this parameter must be 0. It cannot be 1. See the FILEATTR parameter, below.

READ
(input, CHAR, 4) means that the catalog is to be opened for reading.

WRITE
(input, CHAR, 5) means the catalog is opened for writing. WRITE cannot be specified for a directory.

FILEATTR
(input, CHAR, 8) means the catalog is opened to obtain file attribute information about SFS or BFS files. The type of information obtained is determined by the previous parameters in the call:

- Specify GROUP and the storage group number to obtain information about all files in a storage group. (You may not specify storage group number 1, because this is not a user storage group.)
- Specify GROUP and a storage group number of 0 to obtain information about all files in all user storage groups.
- Specify FILESPACE and a file space ID to obtain information about all files in the file space.
- Specify DIRECTORY and *dirname*, *bfsid*, or *namedef* to obtain information about files in a specific directory (but not a subdirectory).

SFS
(input, CHAR, 3) means that CATTYPE=0 catalog entries for BFS objects are returned on subsequent Read Catalog (DMSRDCAT) requests with values expected for SFS objects. This is the default. This parameter is ignored for SFS objects.

BFS
(input, CHAR, 3) means that CATTYPE=0 catalog entries for BFS objects are returned on subsequent Read Catalog (DMSRDCAT) requests with their actual BFS values. This parameter is ignored for SFS objects.

READEXT
(input, CHAR, 7) means the catalog is opened for reading, and extended information is to be returned on subsequent Read Catalog (DMSRDCAT) requests. The extended information is documented under DMSRDCAT.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (READ, WRITE, FILEATTR SFS, FILEATTR BFS, or READEXT).

buffer
(output, CHAR, *length3*) is an output buffer of at least 24 bytes, which upon completion contains the following information:

- Length of the buffer needed, including this field (integer, 4 bytes)
- Token (character, 8 bytes)

  The token identifies this particular open. Specify the token on subsequent Read Catalog, Write Catalog, and Close Catalog routines that you want to relate to this Open Catalog request.
- File pool server release level (integer, 4 bytes)
- ID of server machine (character, 8 bytes)
- Length of the PBN (Physical Block Numbers) Range/minidisk information array (integer, 4 bytes)

If the buffer passed on input is too small, as much data as will fit is placed in the buffer and warning return and reason codes are returned. To get all of the information available, you can then issue a call to DMSCPYBF (SFS Copy Buffer) with the proper buffer size by using the length given in the buffer on output. This is returned only if the Open Catalog was for a storage group or file space.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*). A length of at least 24 must be specified.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used to go to a file pool. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. Before a catalog can be opened for reading or writing, the file space or storage group must be explicitly locked or disabled. The following lists describe the different disabling required to make reading and writing possible.

   No explicit locks are required to open the catalog with intent FILEATTR.

   To get storage group information (GROUP option):

   Intent
   Disable Required

   READ
   Disable Storage Group SHARE or EXCLUSIVE

   READEXT
   Disable Storage Group SHARE or EXCLUSIVE

   WRITE
   Disable Storage Group EXCLUSIVE

   To get file space information (FILESPACE option):

   Intent
   Disable Required

   READ
   None

   READEXT
   None

   WRITE
   Disable File Space EXCLUSIVE

   The DIRECTORY option does not require a disable condition.
2. A request ID is returned if the request is to be asynchronous.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. DMSOPCAT will fail if:
   - Any resource is active in write mode for the work unit.
   - A logical unit of work is in process on the specified resource in the specified work unit, unless a catalog is already open.
6. When you issue DMSOPCAT, the specified resource is registered as a single writer, which means you cannot:
   - Write to another file pool on this work unit.
   - Write to any other protected resource on this work unit.
   - Initiate a protected conversation on this work unit.
7. When the last open catalog is closed for the work unit, the logical unit of work for that resource is implicitly committed, and the resource is no longer active for the specified work unit. However, this commit does not apply to any other active resources in the work unit.

   If, however, an implicit rollback occurs while one or more catalogs are open, the entire work unit will be backed out.
8. Explicit commits (DMSCOMM) are not allowed when one or more catalogs are open. Any attempt to issue an explicit commit will be rejected with a reason code of 79061 (STATE CHECK).

9. All work from the first Open Catalog to the last Close Catalog is treated as an atomic request. That is, CMS will reject the first Open Catalog request if the specified file pool is active for the specified work unit. If, however, the Open Catalog is successful, subsequent Open Catalog requests will be accepted, as will any other request except other atomic functions, and requests which have the Commit option specified. This sequence does not participate in CRR. A noncoordinated commit of the file pool is done by CMS when the last catalog is closed.
10. Directory control directories should not be accessed if you intend to use the block routines (such as DMSOPBLK and DMSRDBLK) in conjunction with the catalog information returned. Having the directory accessed can cause unpredictable results.
11. Some fields in catalogs are used in different ways for BFS file spaces than they are for SFS file spaces. When using the open intent FILEATTR to obtain file attribute information for BFS objects in an SFS environment, you should specify SFS or allow it to default. This ensures that the entries returned on the DMSRDCAT routine are translated into information that is meaningful in an SFS environment. The BFS object information is returned in SFS OBJECTCAT records. If you specify FILEATTR BFS, the returned catalog entries reflect values of how the object is used in a BFS environment. That is, the catalog values are not translated. The BFS object information is returned in BFS OBJECTCAT records.

    Translation applies only to BFS file spaces. If you specify FILEATTR BFS, and the subsequent data returned on DMSRDCAT includes SFS file spaces, no translation is performed on the SFS file spaces. That is, the BFS keyword is ignored.

    There is no translation of the data when the open intent is READ, READEXT, or WRITE.

    For information about the format and content of the catalog records, see "DMSRDCAT - Read Catalog" on page 418.

Return Codes and Reason Codes

For lists of the possible return codes from DMSOPCAT, see Appendix D, "Return Codes," on page 597.

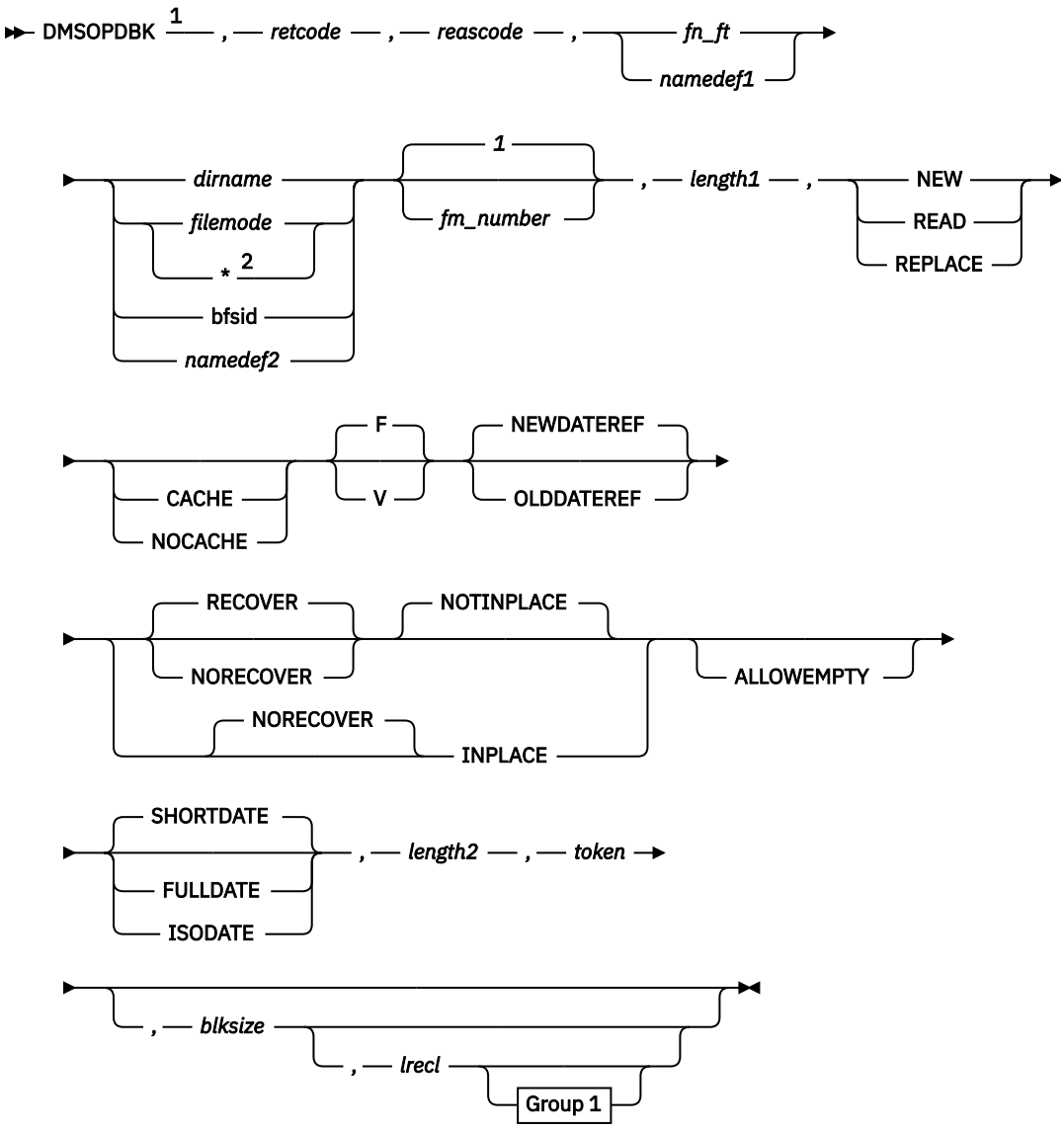
The following table lists the special reason codes returned by DMSOPCAT. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSOPCAT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

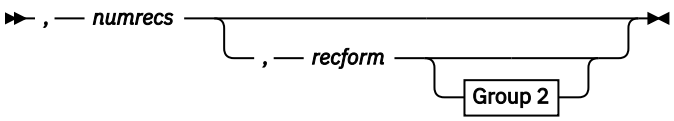
Severity	Reason Code	Description
WARNING	90270	Output buffer was too small to contain all of the requested output. The output has been truncated to the buffer length.
ERROR	30000	You do not have the required file pool administrator authority.
ERROR	32000	Specified file space ID is not enrolled in the file pool.
ERROR	32010	The *filespaceid* part of the file space identifier parameter is missing or is longer than 8 characters.
ERROR	44000	Specified file space does not exist, or you are not the owner of the file space and you do not have file pool administration authority.
ERROR	50100	Specified storage group number is not valid (less than 2 or greater than MAXDISKS server parameter).
ERROR	50103	Storage group number is not valid. Number cannot be 1 or less than zero.
ERROR	50105	Storage group number is not valid. When opening a directory or a file space, the specified storage group number must be zero.

Severity	Reason Code	Description
ERROR	50200	Specified storage group does not exist.
ERROR	56100	Required explicit lock not in effect. Prior to the Open Catalog request, the object being opened must have been locked SHARE or higher for READ, or EXCLUSIVE for WRITE.
ERROR	72000	User has attempted an Open Catalog request for any FILESPACE or DIRECTORY intent and a FILEPOOL RENAME command is currently in process.
ERROR	90210	Extraneous characters in input parameter.
ERROR	90300	Parameter is not valid: FILEATTR format parameter specified incorrectly.
ERROR	90350	Incorrect number of blank-delimited tokens in the *dirname*, *bfsid*, or *namedef* parameter.
ERROR	90415	Length specified for the *wuerror* parameter is not valid. A nonzero length must be at least 12 bytes.
ERROR	90470	Parameter is not valid; must be FILESPACE, GROUP, or DIRECTORY.
ERROR	90472	Requestid is not valid; must be 0 or 1.
ERROR	90485	Buffer length is not valid.
ERROR	90476	File pool ID is not valid.
ERROR	90483	WRITE option is not valid when opening a catalog for a directory.
ERROR	90484	Open type is not valid; must be READ, WRITE, FILEATTR, or READEXT.
ERROR	90500	Directory name is not valid. For BFS file spaces, only the BFS top directory is valid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of *dirname* is longer than 16 characters.
ERROR	90530	The namedef part of the directory name parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was found where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	95400	A logical unit of work is already in process for the specified work unit ID.
ERROR	95500	Intent was WRITE, and you have made uncommitted changes to another file pool for the specified work unit ID.

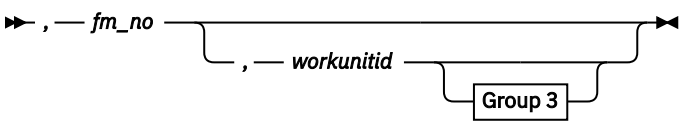
DMSOPDBK - Open Data Block



Group 1



Group 2

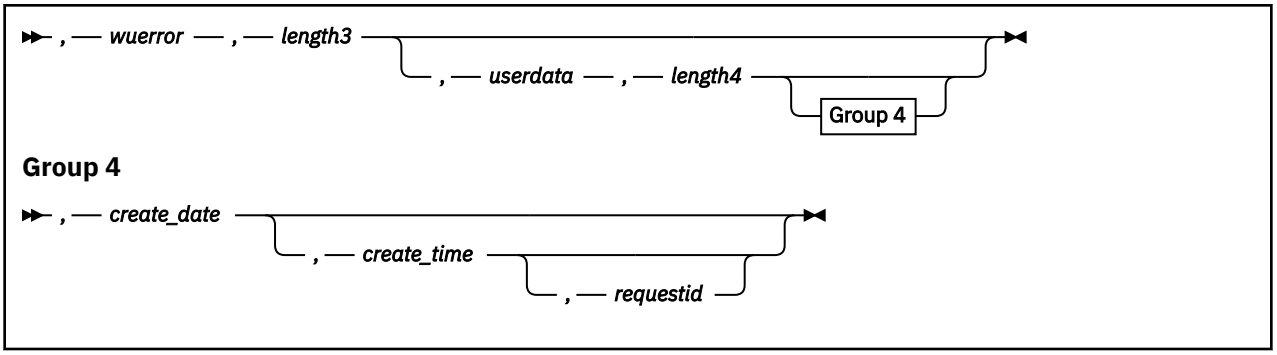


Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

[2] * may be specified for *filemode* only when the open intent is READ.

Group 3



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for invoking a CSL routine is language dependent. DMSOPDBK can be called directly by its name or indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSOPDBK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSOPDBK routine to prepare a file for data block I/O operations. This allows applications to read and write data blocks instead of records as with DMSOPEN. For more information about the data block I/O interface, see the section on Data Block I/O in the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSOPDBK.

reascode
(output, INT, 4) is a variable for the reason code from DMSOPDBK.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file being opened. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the SFS directory that contains the file being opened.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed minidisk or SFS directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode.

An asterisk (*) may be used to specify searching of all accessed minidisks and SFS directories, but only when the intent is READ. If an asterisk is specified, the file modes are searched in their normal A-Z search order. If a file mode letter has an accessed file mode extension, this is also searched.

When the open intent is READ, if the first file matching *fn_ft* or *namedef1* is not a minidisk file or an SFS base or alias file, the file is skipped and searching continues.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system containing the file to be opened.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

fm_number
(input, CHAR, 1) is a variable for specifying the file mode number assigned when a file is opened with an intent of NEW or REPLACE. The default is 1 when a new file is created. If this parameter is omitted for a REPLACE of an existing file, the previous file mode number of the file is used. This parameter is ignored when an existing file is opened with an intent of READ.

If a file mode number is specified for REPLACE, and it is different from the existing file mode number, then the file mode number of the base file and all aliases of that base file are changed to the specified value.

For a BFS file, the only file mode number allowed is 1.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*; plus *dirname*, *filemode*, *bfsid*, or *namedef2*; plus *fm_number*, if specified). See "Compound Variables" on page 15 for coding details.

NEW
(input, CHAR, 3) means to open a new file (one that does not already exist). The file is created and can now be written to.

This intent is not allowed for a BFS file.

READ
(input, CHAR, 4) means that the file can only be read (it must already exist).

REPLACE
(input, CHAR, 7) means that if the file exists, it will be replaced with only the added records. For an SFS file, if the file does not exist, a new file is created. This intent is not allowed for a BFS file if the file does not already exist.

CACHE
(input, CHAR, 5) means the caller intends to read the data primarily sequentially. Specifying this parameter causes the file system to cache several data blocks for the file, performing I/O only when the cache buffer is empty. This generally reduces the number of separate I/O operations performed on the file.

NOCACHE
(input, CHAR, 7) means the caller intends to read the data primarily randomly (not sequentially).

F
(input, CHAR, 1) means fixed (F) length records are being used. This is the default. The format attribute (F or V) is used only when the open intent is NEW or REPLACE.

V
(input, CHAR, 1) means variable (V) length records are being used. The format attribute (F or V) is used only when the open intent is NEW or REPLACE. If the format is not specified, the default is F. The V attribute is not allowed for a BFS file.

NEWDATEREF
(input, CHAR, 10) specifies that SFS should update the date of last reference for the file for any open intent. The date of last reference is the date the file was last read or modified. NEWDATEREF is the default. This parameter is not applicable for a minidisk file.

OLDDATEREF
(input, CHAR, 10) specifies that SFS should not update the date of last reference when the file is opened with an intent of READ. The date of last reference is the date the file was last read or modified. OLDDATEREF is ignored when the file is opened with intents other than READ, because they allow the file to be changed. This parameter is not applicable for a minidisk file.

RECOVER
(input, CHAR, 7) specifies that the file is to be recoverable. When a file is recoverable, uncommitted changes are backed out as the result of an application-initiated rollback or system failure.

The recoverability attribute (RECOVER or NORECOVER) is assigned to the file only when the open intent is NEW or REPLACE. If the recoverability attribute is not specified for one of these intents, a value is assigned. See usage notes "21" on page 326 and "24" on page 326. If an overwrite attribute of INPLACE is specified or assigned, the default recoverability attribute is NORECOVER.

When an alias is opened for REPLACE, the recoverability of the base file and all aliases on the base are set to RECOVER.

This parameter is not applicable for a minidisk file. All minidisk files are nonrecoverable, and you are warned if RECOVER is specified.

NORECOVER
(input, CHAR, 9) specifies that changes to the file are not rolled back in the event of an application-initiated rollback. In most cases, the updates are committed.

The recoverability attribute (RECOVER or NORECOVER) is assigned to the file only when the open intent is NEW or REPLACE. If the recoverability attribute is not specified for one of these intents, a value is assigned. See usage notes "21" on page 326 and "24" on page 326. If an overwrite attribute of INPLACE is specified or assigned, the default recoverability attribute is NORECOVER.

When an alias is opened for REPLACE, the recoverability of the base file and all aliases on the base are set to NORECOVER.

This parameter is not applicable for a minidisk file. All minidisk files are nonrecoverable, and you are warned if NORECOVER is specified.

This attribute is not allowed for a BFS file.

NOTINPLACE
(input, CHAR, 10) specifies that readers see a consistent version of the file from Open to Close.

The overwrite attribute (NOTINPLACE or INPLACE) is assigned to the file only when the open intent is NEW or REPLACE. If an overwrite attribute is not specified for one of these intents, a value is assigned. See usage notes "21" on page 326 and "24" on page 326.

If the specified file is an alias on an Open for REPLACE, then the overwrite attribute of the base and all aliases on the base, including the one specified, will be set to NOTINPLACE.

This parameter is not applicable for a minidisk file. All minidisk file mode numbers from 0-5 are treated as NOTINPLACE. A warning will result if NOTINPLACE is specified for a minidisk file and the file mode number is 6.

INPLACE
(input, CHAR, 7) specifies that updates are to be made in place where possible. This can reduce DASD utilization and enables readers to see file updates by a concurrent writer without closing and reopening the file. Users that have an INPLACE file open for read have to reopen the file to see extensions (new records or blocks) that have been written and committed to the file.

The overwrite attribute (NOTINPLACE or INPLACE) is assigned to the file only when the open intent is NEW or REPLACE. If an overwrite attribute is not specified for one of these intents, a value is assigned. See usage notes "21" on page 326 and "24" on page 326.

If the specified file is an alias on an Open for REPLACE, then the overwrite attribute of the base and all aliases on the base, including the one specified, will be set to INPLACE.

This parameter is not applicable for a minidisk file. All minidisk files with a file mode number of 6 are treated as INPLACE. A warning will result if INPLACE is specified for a minidisk file and the file mode number is not 6.

This attribute is not allowed for a BFS file.

ALLOWEMPTY
(input, CHAR, 10) indicates that a file with no records may be created as a result of this DMSOPDBK request. This may allow either a new empty file to be created or an existing file to be replaced with an empty file, if there are no records in the file at the time a DMSCLDBK or COMMIT is issued.

This parameter is not applicable for a minidisk file. Empty files are not allowed on minidisks. A warning will result if ALLOWEMPTY is specified for a minidisk file.

SHORTDATE
(input, CHAR, 9) indicates the format of the *create_date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *create_date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *create_date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (NEW, READ, or REPLACE; plus CACHE or NOCACHE, if specified; plus F or V, if specified; plus NEWDATEREF or OLDDATEREF, if specified; plus RECOVER or NORECOVER, if specified; plus NOTINPLACE or INPLACE, if specified; plus ALLOWEMPTY, if specified; plus SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

token
(output, CHAR, 8) is a variable for returning a value that uniquely identifies the file being opened. You specify the token on subsequent Read Data Block (DMSRDDBK), Write Data Block (DMSWRDBK), and Close Data Block (DMSCLDBK) routines related to this Open.

blksize
(output, INT, 4) is a variable containing the block size of the file.

lrecl
(input/output, INT, 4) is a variable containing the logical record length of the file.

When the intent is READ, this parameter is not required. If it is specified, it returns the actual record length of the file.

When the intent is NEW or REPLACE, this parameter is unchanged.

numrecs
(input/output, INT, 4) is a variable containing the number of records in the existing file.

When the intent is READ, this parameter is not required. If it is specified, it returns the actual number of records in the file.

When the intent is NEW or REPLACE, this parameter is unchanged.

A value of 0 is returned if the file is empty.

recform
(output, CHAR, 1) is a variable in which a value is returned that indicates the record format of the file:

F
indicates all records have the same length

V
indicates the records may have varying lengths

fm_no
(output, CHAR, 1) is a variable containing the file mode number of the file.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and will result in an error return code.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

create_date
(input, CHAR, 8 or 10) is a variable for specifying the date-of-creation attribute that will be saved with the file. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years. SHORTDATE is the default, which is the 2-digit year format.

This parameter is applicable only if the file is a new SFS file. (That is, the file did not exist prior to the execution of DMSOPDBK.) If it is specified when you open the file with an intent of REPLACE, a warning is issued.

If the file is new, you can have the system determine the creation date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. If you do not specify the SHORTDATE, FULLDATE, or ISODATE parameter, and you omit the *create_date* variable, or leave blanks, the system generates the creation date. Note that the system uses Coordinated Universal Time (UTC) when it determines the creation date.

If you specify a date, use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

create_time
(input, CHAR, 8) is a variable for specifying the time-of-creation attribute that will be saved with the file.

This parameter is applicable only if the file is a new SFS file. (That is, the file did not exist prior to the execution of DMSOPDBK.) If it is specified when you open the file with an intent of REPLACE, a warning is issued.

If the file is new (did not exist prior to DMSOPDBK), you can have the system determine the creation time by omitting the parameter or by specifying 8 blanks. Note that the system uses Coordinated Universal Time (UTC) when it determines creation time.

If you specify a time, a colon (:) must be used as the separator character and 2 digits must be specified for each position in the form *hh:mm:ss*, where *hh* is the hour, *mm* is the minutes, and *ss* is the seconds. For example, you would render 9 minutes and 15 seconds after 3 pm as `15:09:17` in the *create_time* parameter.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid* and is passed on a later Check (DMSCHECK) request.

The processing of a given DMSOPDBK request may not require communication with the file pool server. In this case, the operation is performed synchronously regardless of the value specified in *requestid* and the value of the *requestid* parameter is not changed.

Usage Notes

1. If an SFS or BFS file is opened for REPLACE and closed before any blocks are written, the file remains unchanged, unless ALLOWEMPTY is also specified.
2. If a minidisk file is opened for REPLACE and closed before any blocks are written, the file is erased.
3. If an SFS or minidisk file is opened for REPLACE and the file does not exist, the open is accepted. If the record format (F or V) is not specified, it defaults to fixed (F).
4. If an SFS file is opened for REPLACE and the file does not exist, or if a file is opened for NEW, it is not visible to other applications until the file is written to and committed.
5. There is no default choice for caching. If neither CACHE nor NOCACHE is specified, the system chooses the more appropriate method.
6. Using COMMIT when your SFS or BFS files are open (using DMSOPDBK) will not result in the updates being written and made available to readers. The updates can only be committed after the file is closed.
7. DMSOPDBK will allow a CMS file to be opened more than once by an application. The Shared File System offers better sharing and data integrity than minidisks. The following conditions apply:
   - An SFS or BFS file can be opened more than once for READ and at most once for NEW or REPLACE.
   - A minidisk file can be opened more than once for READ or once for NEW or REPLACE. Once the file has been opened for NEW or REPLACE, it must be closed before it can be reopened.
8. When CMS subset mode is entered, the file system uses the Close Data Block routine (DMSCLDBK) to close any files opened with DMSOPDBK. In variable format files opened for output, closing the file deletes data past the last complete record. After the return from subset mode, the file is not reopened, and the token previously returned by DMSOPDBK is no longer useable.
9. If the file is fixed record format, all records in the data blocks are of the same length. If the file is variable format, each record in the data blocks is composed of a 2-byte record length, followed by the actual data itself (not including the 2 length bytes). Variable length records should not be modified using the data block interface. If incorrect values are placed in the file, the records in the file can be stored in an unpredictable format. Use the record interface (DMSOPEN, DMSREAD, DMSWRITE, and DMSCLOSE) to modify existing files for both fixed and variable format files.
10. The types of operations that can be performed on the file are restricted to the open intent of the file. When the intent is READ, the file may not be written to, unless a separate DMSOPDBK or DMSOPEN is used. When the intent is NEW or REPLACE, the file may be only be written to, not read.
11. You will receive an error when attempting to use the Open Data Block routine with the READ parameter on a file that does not exist.
12. If you choose the NEW or REPLACE option and a file or directory is locked, the open will fail. An exception is when the issuer has an UPDATE or EXCLUSIVE lock on the file or directory.
13. If you choose the READ option and a file or directory is locked, the open is allowed. An exception is when another user has an EXCLUSIVE lock on the file or directory. In this case the open is not allowed.

14. When reading files that reside in directory control directories, remember that:
    - You do not see changes made to a directory during a period that you have it accessed read-only. To see changes, you must reaccess the directory.
    - If you do not have the directory accessed, you must close and reopen the file to see any changes made to the file after you opened it.

    Exception: You can see changes to INPLACE files without reopening the file or reaccessing the directory.
15. You cannot open a file with the NEW or REPLACE option if:
    - Another user has the file or directory locked
    - The file resides in a directory control directory, and another user has the directory accessed R/W
    - The file resides in a directory control directory, and you have the directory accessed R/O
    - (Applies only to REPLACE option) the file is an alias residing in a file control directory accessed R/W, and its base file resides in a directory control directory that is accessed read-only
    - The file resides on a minidisk and the file mode is accessed as R/O
    - The file resides on a minidisk and is already opened for output using the FSOPEN interface.
16. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file you are opening. If the file being opened is on a minidisk, *userdata* is not used.
17. If you specify OLDDATEREF or NEWDATEREF and execute the routine against a file pool that does not support dates of last reference, the parameters are ignored.
18. File pool servers that are not at the current release level may not support the creation date and creation time attributes. Those servers ignore *create_date* or *create_time*.
19. In general, you cannot see another user's uncommitted changes. If a file opened for READ has the INPLACE attribute, this is not true. Because these files are updated-in-place, it is possible that a reader may see a writer's updates without closing the file or committing the data. Because CMS buffers the file's data, the writer's updates must first be written to DASD and the reader's buffers must then be read from DASD before the reader sees these updates. However, the timing of when the buffers get read or written depends on the file size, the caching options specified when the file was opened, and the record access patterns of both the reader and the writer.
20. Although you generally cannot see another user's uncommitted changes, you can see your own uncommitted changes on the same work unit. If you update and close a file and then reopen it on the same work unit, you will see the updated version. If you do not close the file and open it a second time, or if you open it on another work unit, you only see the last committed version of the file.
21. If you do not specify recoverability and overwrite attributes for a file that is being created by this DMSOPDBK, then the attributes given to the file will be based on the defaults for files with that file mode number as established by the DMSPUSHA (Push Attributes) routine. If a Push Attributes has not been done, then the recoverability file attribute defaults to RECOVER, and the overwrite attribute defaults to NOTINPLACE.
22. The INPLACE and NORECOVER attributes are not given to a file until it is closed and committed, so that the file system can recover partially written files in case of an error.
23. If the overwrite and recoverability specification on DMSOPDBK READ does not match that of the base file, a warning is issued.
24. If the file mode number, recoverability, and overwrite attributes are specified for a DMSOPDBK REPLACE, the resulting file uses the new attributes as specified. Otherwise, the attributes are inherited from the file being replaced, if one exists. The defaults set by DMSPUSHA (Push Attributes) do not apply when replacing an existing file.
25. If an existing file is in DFSMS/VM migrated status and is opened with intent of READ, the file is recalled unless RECALL has been set to OFF. Opening for Replace does not trigger a recall.

26. If a file is in DFSMS/VM migrated status, an asynchronous DMSOPDBK request with an intent of READ causes the file to be recalled. Even if the work unit is rolled back before DMSOPDBK processing finishes, the file is recalled.
27. When *wuerror* is specified on DMSOPDBK and an SFS error occurs during the recall process, FPERROR is updated to include the specific reason code of the failing request and the file pool ID of the failing resource.
28. For an asynchronous request, a return code is given indicating whether the request was accepted for processing or was immediately rejected.
29. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
30. Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.
31. BFS files may contain more than 2^{31}-1 records (bytes) when created by means other than CMS record interfaces. When opening a BFS file with CMS record interfaces, the open attempt will fail if the file has more than 2^{31}-1 records. An attempt to write more than 2^{31}-1 records will also fail.
32. Opening a BFS file using this interface requires file pool administration authority.
33. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *create_date* parameter.
34. When *create_date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
35. When *create_date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
36. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSOPDBK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSOPDBK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSOPDBK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	10220	File mode value other than 1 specified for a BFS file.
WARNING	44030	Intent was REPLACE, and the file did not previously exist.
WARNING	44035	Intent was REPLACE, and the file did not previously exist. In addition, ALLOWEMPTY was specified but the file pool or minidisk does not support this option.

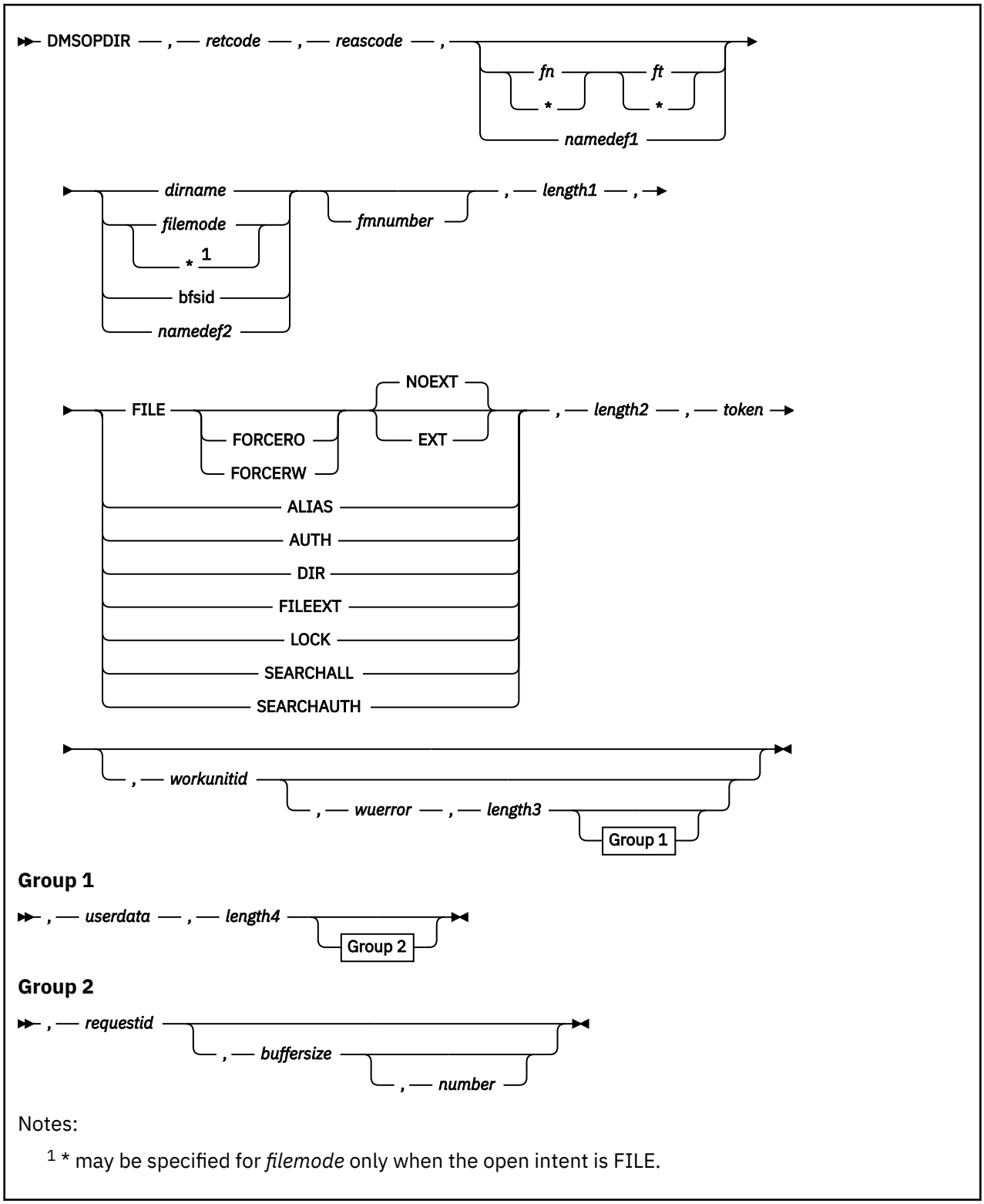
Severity	Reason Code	Description
WARNING	51051	You specified nonblank values for *create_date* or *create_time* and opened an existing file with an intent of REPLACE. You cannot change the creation date or creation time of an existing file.
WARNING	65900	A system error occurred in the file pool server access routine.
WARNING	78107	Recoverability attribute does not match that of the base file. The NORECOVER attribute is not allowed for a BFS file.
WARNING	78108	Overwrite attribute does not match that of the base file. The INPLACE attribute is not allowed for a BFS file.
WARNING	90330	Parameter not valid.
WARNING	90620	RECOVER option was specified for a minidisk file. All minidisk files are nonrecoverable.
WARNING	90621	INPLACE option was specified for a minidisk file with a file mode number other than 6 or NOTINPLACE was specified for a minidisk file with a file mode number of 6.
WARNING	90622	ALLOWEMPTY option was specified for a minidisk file.
WARNING	98700	Server managing this file does not support certain parameters of DMSOPDBK, and takes these actions when it encounters them: **NORECOVER** RECOVER is assumed. **INPLACE** NOTINPLACE is assumed. **ALLOWEMPTY** The server ignores the parameter. ***create_date*** The server ignores the parameter. ***create_time*** The server ignores the parameter.
ERROR	20000	Intent was NEW, but the file already exists.
ERROR	31000	Mixing recoverable and nonrecoverable work for the same file is incorrect within a single work unit.
ERROR	30500	System error. The combination of attributes INPLACE and RECOVER is not supported.
ERROR	44000	• Intent was NEW, and the directory does not exist or you are not authorized to create a file in the directory, **or**, • Intent was REPLACE, and the file does not exist and you are not authorized to create a file in the directory, or you are not authorized to write to the file, or the directory does not exist, **or**, • Intent was READ, and the file does not exist or you are not authorized to read the file.

Severity	Reason Code	Description
ERROR	44200	Your attempt to open a file failed because you already have it open with DMSOPEN, DMSOPBLK, or DMSOPDBK: • You tried to open an SFS or minidisk file for NEW or REPLACE and you already have it opened for NEW, WRITE, or REPLACE. • Or you tried to open a minidisk file for REPLACE and you already have it opened for READ. • Or you tried to open a minidisk file for READ and you already have it opened for NEW, WRITE, or REPLACE. This reason code is returned for SFS files only if both attempts to open the file occurred on the same work unit.
ERROR	50500	Attempt to exceed the maximum number of 4KB file space blocks allowed for this user. Applicable only when NEW or REPLACE option is specified.
ERROR	63700	Directory control directory specified and another user has it locked or accessed R/W, or you have it accessed R/O.
ERROR	64300	The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.
ERROR	65200	File is migrated and implicit RECALL is set to OFF. Set RECALL ON or issue a DFSMS RECALL command.
ERROR	65400	Specified operation cannot be performed on an external object. Request cannot be performed on a BFS object due to one or more of the following: • Record format other than F specified for a BFS file. • Intent was not READ or REPLACE for a BFS file. • BFS file contains more than 2^{31}-1 records (bytes). • Intent was REPLACE for a BFS file and the file does not exist.
ERROR	65500	Input file is migrated and there is no active SMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)
ERROR	65900	System error. A server error occurred during an attempt to open a file for a purpose other than reading; that is, with NEW or REPLACE.
ERROR	66100	Input file has been moved and DFSMS/VM recall processing has been disabled.
ERROR	71200	The server encountered an error when it attempted to access a file.
ERROR	81058	System error. An addressing exception occurred while accessing data in a data space owned by a file pool server machine.
ERROR	90105	Incorrect record format.
ERROR	90250	The file name and file type (or *namedef*) are required but were not specified.
ERROR	90300	Incorrect intent parameter. Valid intents are NEW, READ, or REPLACE.

Severity	Reason Code	Description
ERROR	90307	*lrecl* must not be negative.
ERROR	90310	Incorrect option specified. Valid options are CACHE or NOCACHE, F or V, OLDDATEREF or NEWDATEREF, RECOVER or NORECOVER, INPLACE or NOTINPLACE, ALLOWEMPTY, SHORTDATE or FULLDATE or ISODATE.
ERROR	90315	The open intent (NEW, READ, or REPLACE) was not specified.
ERROR	90320	Conflicting options in CSL parameter list. NEW, READ, and REPLACE are mutually exclusive parameters, CACHE and NOCACHE, if specified, are mutually exclusive parameters, F and V, if specified, are mutually exclusive parameters, NEWDATEREF and OLDDATEREF, if specified, are mutually exclusive parameters, RECOVER and NORECOVER, if specified, are mutually exclusive parameters, INPLACE and NOTINPLACE, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: NEW, READ, REPLACE, CACHE, NOCACHE, F, V, NEWDATEREF, OLDDATEREF, RECOVER, NORECOVER, INPLACE, NOINPLACE, ALLOWEMPTY, SHORTDATE, FULLDATE, and ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least two and not more than four tokens in the string.
ERROR	90410	Incorrect length specified for one of the character variables.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	*requestid* must be 0 or 1.
ERROR	90484	WRITE is not a valid Open Data Block type. Must be NEW, READ, or REPLACE.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Severity	Reason Code	Description
ERROR	90496	The date specified is incorrect; it must be a number 0-9.
ERROR	90498	The time specified is in incorrect format; it must be in the format *hh:mm:ss*.
ERROR	90499	The time specified is incorrect; it must be a number 0-9.
ERROR	90500	The specified directory name is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.
ERROR	90602	Incorrect file mode specified. Asterisk only allowed for open intent of READ.
ERROR	90603	Attempted to open the file for output and file mode is read-only.
ERROR	90604	Minidisk file already open using the FS macro interface.
ERROR	90605	An attempt was made to associate an ACF with a minidisk file.
ERROR	90606	I/O error found when processing a minidisk file.
ERROR	90680	I/O error accessing OS dataset.
ERROR	90681	OS read password protected dataset.
ERROR	90682	OS dataset organization is not BSAM, QSAM or BPAM.
ERROR	90683	OS dataset more than 16 extents.
ERROR	90684	Attempt to open a file on an OS or DOS formatted minidisk.
ERROR	90685	Received an unexpected return code while opening a minidisk file.

DMSOPDIR - Open Directory



Notes:

[1] * may be specified for *filemode* only when the open intent is FILE.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSOPDIR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSOPDIR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSOPDIR routine to open an SFS directory, a byte file system, or a minidisk that is to be read later using any of the Get Directory routines.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSOPDIR.

reascode
(output, INT, 4) is a variable for the reason code from DMSOPDIR.

fn ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of files in the directory being opened. For a BFS file, these are system-generated values. You can specify an asterisk (*) for either *fn* or *ft* to indicate all file names or all file types.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the SFS directory being opened.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed minidisk or SFS directory to be opened. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode.

An asterisk may be used to specify searching of all accessed minidisks and SFS directories, but only if the intent is FILE.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system (BFS top directory) to be opened.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number. Only files with this file mode number will be passed on a later Get Directory. If not specified, all file mode numbers will be returned.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameters (*fn_ft* or *namedef1*, if specified; plus *dirname*, *filemode*, *bfsid*, or *namedef2*; plus *fmnumber*, if specified). See "Compound Variables" on page 15 for coding details.

FILE
(input, CHAR, 4) opens a directory for which a later Get Directory (DMSGETDF or DMSGETDI) will get information. You must specify the *fn_ft* or *namedef1* parameter on input. Information returned from a later Get Directory will include:

File name

File type
File mode
File mode number
Record format
Logical record length
Number of records
Number of blocks
Date of last update
Time of last update
Owner
Type (directory, alias, base, erased, revoked)
Authority
Directory attribute

You can also open a minidisk "directory" using FILE. This will search for files on the minidisk specified by *filemode*.

For External Security Manager (ESM)-protected objects, a flag indicating that it is ESM-protected is returned, not the authorities. For the exact output format, see the description of the Get Directory routine (DMSGETDI).

This intent is not allowed for BFS.

FORCERO
(input, CHAR, 7) establishes a read-only status for a directory during the time it is open. This read-only status has no effect on file control directories, but does affect the use of directory control directories (see the Usage Notes). FORCERO can be specified only when the intent is FILE.

FORCERW
(input, CHAR, 7) establishes a read/write status for a directory during the time it is open. This read/write status has no effect on file control directories, but does affect the directory control directories (see the Usage Notes). FORCERW can be specified only when the intent is FILE.

EXT
(input, CHAR, 3) indicates that file mode extensions will be searched. When a *filemode* is specified and the intent is FILE, file searching will be done on the minidisk or SFS directory accessed at *filemode* and will continue on any file mode extension.

NOEXT
(input, CHAR, 5) indicates that file mode extensions will not be searched. This may only be specified with intent of FILE. If neither EXT or NOEXT is specified, NOEXT will be used as the default.

ALIAS
(input, CHAR, 5) opens a directory for which a later Get Directory (DMSGETDL or DMSGETDI) will get information. You must specify the *fn_ft* or *namedef1* parameter. Information returned from a later Get Directory consists of: file name, file type, file mode number, format, logical record length, records, blocks, date, time, owner, type (alias, base, erased, revoked). For type=alias, the owner of the base file, base file name, file type, and directory name are also returned. The last three fields will be blank if you are not authorized for the directory. For the exact output information, see the Get Directory routine (DMSGETDI). For type=base, the owner of the alias, alias file name, file type, and directory name are also returned. The last three fields may be blank if you are not authorized for the directory. In this case, the number of aliases to the base file is returned instead.

You must have authority on the file name specified. If you use an ambiguous file ID, you must have authority on the directory.

AUTH
(input, CHAR, 4) opens a directory for which a later Get Directory (DMSGETDT or DMSGETDI) will get information about authorizations that the caller has or has granted to others.

You can specify the *fn_ft* or *namedef1* parameter. Information returned from a later Get Directory (DMSGETDI) consists of: file name, file type, file mode number, type (base, alias, directory, erased,

revoked), grantee, authority, and the directory attribute. The ESM-protected flag is also returned. If it is on, authorities are returned. However, these are the base authorities, not the ESM-held ones. For the exact output information, see the Get Directory routine (DMSGETDI).

You must have authority on the file name specified. If special characters (* or %) are specified, you must also have authority on the directory.

This intent is not allowed for BFS.

DIR
(input, CHAR, 3) opens a directory for which a later Get Directory (DMSGETDD or DMSGETDI) will get information. You must not specify the *fn_ft* or *namedef1* parameter.

Calls to Get Directory return information about the directory and any subdirectories to which you have authority. For the exact output information, see the Get Directory routine (DMSGETDI) or the Get Directory - Dir routine (DMSGETDD).

FILEEXT
(input, CHAR, 7) opens a directory for which a later Get Directory (DMSGETDX or DMSGETDI) request will return extended information. You must specify *fn_ft* or *namedef1* when you use FILEEXT. If you open a directory with FILEEXT, Get Directory returns all of the file attributes returned for FILE (except file mode), plus:

File space type (SFS or BFS)
Date of last reference
Date of creation
Time of creation
Recoverability
Overwrite
Date of last change
Time of last change

The extended attributes are returned only when FILEEXT is specified. For best performance, use FILE unless you need the extended attributes.

For the exact output format, see the description of the Get Directory routine (DMSGETDI) or the Get Directory - File Extended routine (DMSGETDX).

LOCK
(input, CHAR, 4) opens a directory for which a later Get Directory (DMSGETDK or DMSGETDI) routine will get information. You can specify the *fn_ft* or the *namedef1* parameter. Information returned from a later Get Directory consists of: file name, file type, file mode number, type (base, alias, directory, erased, revoked), lock holder, and type of lock. For the exact output information, see the Get Directory routine (DMSGETDI).

SEARCHALL
(input, CHAR, 9) opens a directory for which later Get Directory calls (DMSGETDA or DMSGETDI) will get information. Information is made available only for objects that match the specified file ID in the specified directory and all its subdirectories.

DMSOPDIR searches only SFS subdirectories:

- For which you have read or write authority
- And which are not locked EXCLUSIVE by another user.

You must specify the *fn_ft* or *namedef1* parameter on input. Information returned from a later Get Directory consists of: file name, file type, file mode number, format, logical record length, records, blocks, date, time, directory name, owner, type (directory, alias, base, erased, revoked), and the directory attribute. For the exact output information, see the Get Directory routine (DMSGETDI).

This intent is not allowed for BFS.

SEARCHAUTH
(input, CHAR, 10) is the same as SEARCHALL except that a later Get Directory (DMSGETDS or DMSGETDI) will only return information for files and subdirectories that the issuer is authorized for.

DMSOPDIR only searches SFS subdirectories:

- For which you have read or write authority
- That are not held by another user with an EXCLUSIVE lock.

For the exact output information, see the Get Directory routine (DMSGETDI).

This intent is not allowed for BFS.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (FILE, and optionally FORCERO or FORCERW and EXT or NOEXT; or ALIAS, AUTH, DIR, FILEEXT, LOCK, SEARCHALL, or SEARCHAUTH). See "Compound Variables" on page 15 for coding details.

token
(output, CHAR, 8) is a variable for returning a value that identifies an instance of an open directory. Pass the value of this variable on future calls to the Get Directory and Close Directory (DMSCLDIR) routines related to this Open Directory.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

If the returned request ID is still 1, no server call was needed, and it is not necessary to call DMSCHECK, because the function has already been completed.

buffersize
(output, INT, 4) is the length of the directory entry information associated with the specified intent keyword. Although the amount of directory entry information can be greater then 16KB, the maximum value that can be returned in *buffersize* is 16K + 1 (16 385 or X'0401'), which indicates that there is more than 16KB of directory information.

If a file is not specified or if the file ID is ambiguous, DMSOPDIR returns the total length of all the entries for the directory. If a file is specified, DMSOPDIR returns the length of all the directory entries associated with the file. If the intent is FILE, DMSOPDIR returns a buffersize of zero.

number
(output, INT, 4) is the number of directory entries associated with the specified directory. If there is more than 16KB of directory information, *number* contains the number of entries that fit into 16KB.

If a file is not specified or if the file ID is ambiguous, DMSOPDIR returns the number of entries for the directory. If a file is specified, the DMSOPDIR returns the number of entries associated with the file. If the intent is FILE, DMSOPDIR returns a number of zero.

Usage Notes

1. No implicit locks are held across the Open Directory (DMSOPDIR), Get Directory (DMSGETDD, DMSGETDF, DMSGETDI, and so on), and Close Directory (DMSCLDIR) sequence. This means that while a directory is open, changes to information in the directory may or may not be reflected in subsequent Get Directory requests.

   When a directory is open, 16KB of data is sent to the user's virtual machine for use by Get Directory. If changes are made while the directory is open, (for example, a file is erased) those changes are not reflected in the data in the user machine. However, if Get Directory needs to get another 16KB of data from the server, the changes may be reflected in the new set of data.

   An exception to this rule is when the directory is open for intent FILE. In this case, changes are immediately reflected and subsequent Get Directory requests reflect those changes. For example, if the directory is opened for intent FILE and a file in the directory is erased, subsequent Get Directory requests will not return an entry for that file. An exception to this exception is when the intent is FILE and the work unit used was gotten for another user ID (see DMSGETWU for more details). In this case, changes are not immediately reflected and subsequent Get Directory calls operate that same as intents other than FILE.
2. You cannot open an empty directory with intent FILE or FILEEXT.
3. You can open a minidisk directory for FILE to get information on specific files on that minidisk. See the Get Directory function for details on how the information differs between files on SFS directories and CMS minidisks.
4. If you are opening a directory control directory, use the FORCERW or FORCERO parameters to get the level of sharing you need. If you intend only to read the directory or its files during the time you have the directory open, specify FORCERO. If, however, you intend to write to the directory or its files, specify FORCERW.

   When you omit FORCERO and FORCERW, CMS uses directory ownership to determine whether you will be allowed to read or write. (CMS follows the same rules it follows for the ACCESS command.) If you own the directory, for example, CMS will try to establish a read/write status for you. If you do not own the directory, CMS, by default, establishes a read-only status.

   Letting CMS establish a read/write status for you when you intend only to read will prevent other authorized users from writing to your directory control directory. Conversely, letting CMS establish a read-only status for you will prevent you from updating the directory control directory.

   The FORCERO and FORCERW parameters have no effect on the sharing of file control directories.
5. You can specify the FILEEXT intent even if you are not sure whether the server managing the directory supports all of the extended file attributes. In this case, a 1 (for RECOVER) is returned for the recoverability attribute and zeros are returned for the remaining extended attributes on subsequent Get Directory requests. If you specify FILEEXT intent when the server managing the directory does not support the date of last reference attribute, however, you will receive an error message. If the server supports the date of last reference attribute but one has not been established for the file, zeros will be returned.
6. When AUTH is specified, later Get Directory requests will find directories for which you have only NEWREAD or NEWWRITE authority (not READ or WRITE authority). For all other open intents,

directories for which you have only NEWREAD or NEWWRITE authority are *not found* (reason code 44000 is returned).

7. If your installation does not use an external security manager (ESM), do not specify the *userdata* parameter.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the directory you are opening. If the directory being opened is minidisk, the *userdata* is not used.
8. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
9. If you have an active asynchronous request for a file pool, you cannot issue other requests (except a rollback request) for the affected file pool on that same work unit.
10. All minidisk requests are done synchronously.
11. The *buffersize* and *number* output variables can be used by an application on a DMSGETDI request. Use *buffersize* as the length of *buffer* (*length1*) and *number* as the number of entries to be returned by DMSGETDI (*number*). However, if an ambiguous file ID is used on the DMSOPDIR request, the values in *buffersize* and *number* are for all the entries that fit into 16KB.
12. For a BFS file space, only intents of FILEEXT, LOCK, ALIAS, and DIR are supported. File pool administration authority is required. Only information for BFS regular files is returned.
13. Opening a directory with intent FILE causes CMS to access the SFS directory implicitly.

Return Codes and Reason Codes

For lists of the possible return codes from DMSOPDIR, see Appendix D, “Return Codes,” on page 597.

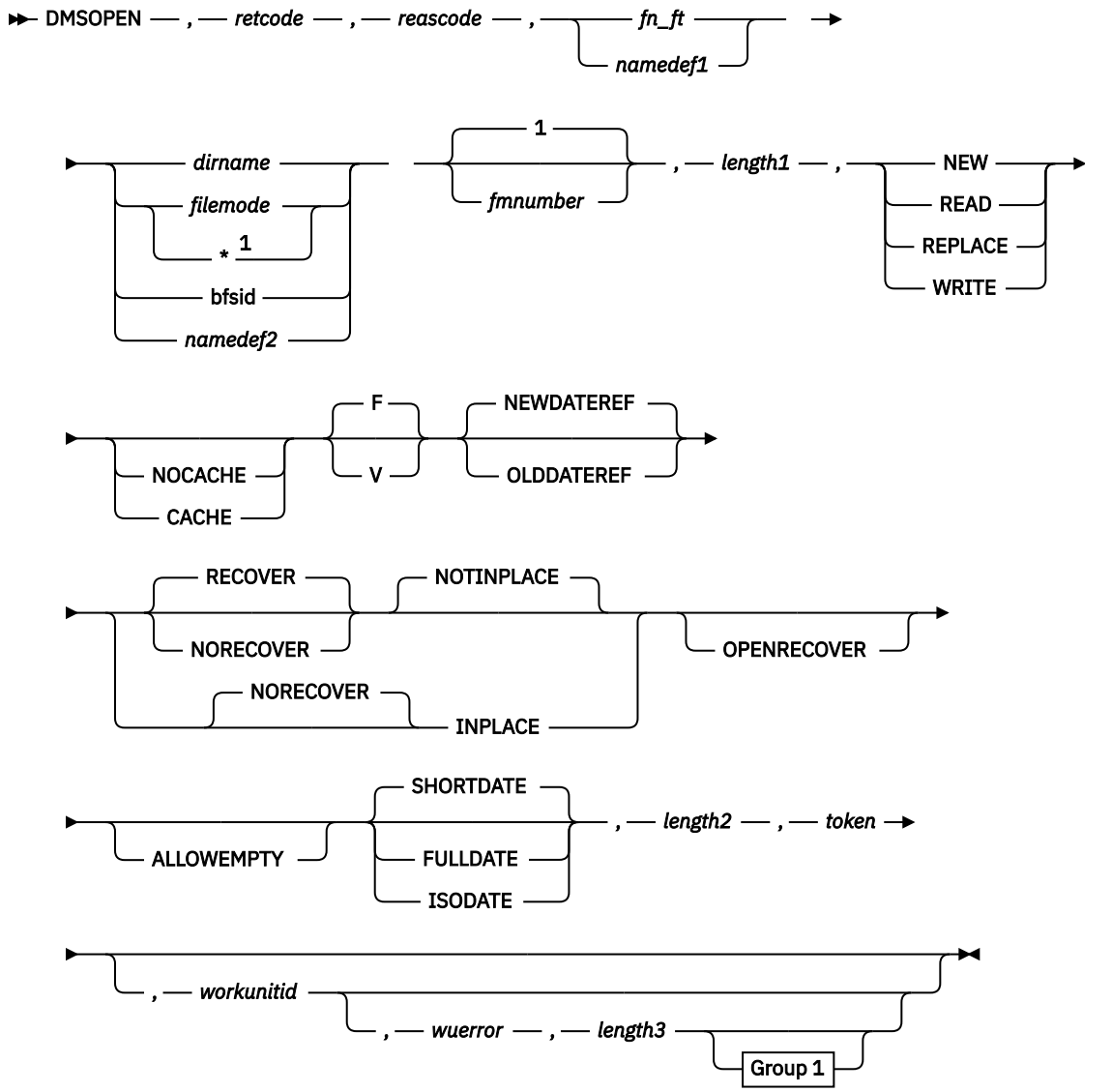
The following tables lists the special reason codes returned by DMSOPDIR. ERROR means the request failed. Errors cause return code 8 or 12.

DMSOPDIR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	44000	For Open Directory for AUTH, LOCK, ALIAS, or FILEEXT, the specified directory or specified file does not exist or you are not authorized to read the directory.
ERROR	64200	The directory control directory you specified is empty. It has no files or subdirectories.
ERROR	65400	The operation cannot be performed against a BFS object.
ERROR	81058	Server failure. The specified directory was placed in a data space, but your authorization to that data space was revoked because of server failure. To regain your authorization to the data space, simply reexecute the routine when the server is available. (The server internally handles data space authorizations based on your authority to the directory.)
ERROR	90220	The specified file does not exist in the specified directory, or no files exist that match the specified wildcard pattern.
ERROR	90230	The specified directory does not exist or you are not authorized to it.
ERROR	90250	File ID must be specified for Open Directory for SEARCHALL, SEARCHAUTH, ALIAS, FILE, or FILEEXT.

Severity	Reason Code	Description
ERROR	90255	File ID cannot be specified for Open Directory for DIR.
ERROR	90300	Invalid parameter specified. Type of Open Directory is invalid, extraneous keywords are present, or FORCERO/FORCERW specified without intent FILE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.
ERROR	90410	Invalid length specified for one of the character variables.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Directory is on a CMS minidisk and open intent was not FILE.

DMSOPEN - Open



Notes:

[1] * may be specified for *filemode* only when the open intent is READ.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSOPEN is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSOPEN
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSOPEN" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSOPEN routine to make a file usable for subsequent reading or writing of data records.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSOPEN.

reascode
(output, INT, 4) is a variable for the reason code from DMSOPEN.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file being opened. For a BFS file, these are system-generated values.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the name of the SFS directory that contains the file to be opened.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed minidisk or SFS directory to be searched. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode.

An asterisk (*) may be used to specify searching of all accessed minidisks and SFS directories, but only when the file is opened for reading. In this case, the search finds the first minidisk file or SFS base file or alias that matches *fn_ft* or *namedef1*. If an asterisk is specified, the file modes are searched in their normal A-Z search order. If a file mode letter has an accessed file mode extension and the file is being opened for reading, the extension is also searched.

bfsid
(input, CHAR, 1-18) is a variable for specifying the name of the byte file system containing the file to be opened.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname*, *filemode*, or *bfsid*.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number assigned when a file is opened with an intent of NEW, REPLACE, or WRITE (for a new file). The default is 1 when a new file is created. If this parameter is omitted for a REPLACE of an existing file, the previous file mode number of the file is used. This parameter is ignored when an existing file is opened with an intent of READ or WRITE.

If a file mode number is specified for REPLACE, and it is different from the existing file mode number, then the file mode number of the base file and all aliases of that base file are changed to the specified value.

For a BFS file, the only file mode number allowed is 1.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*; plus *dirname*, *filemode*, *bfsid*, or *namedef2*; plus *fmnumber*, if specified). See "Compound Variables" on page 15 for coding details.

NEW
(input, CHAR, 3) means to open a new file (one that does not already exist). The file is created and can now be written to and read from. This intent is not allowed for BFS.

READ
(input, CHAR, 4) means that the file can only be read (it must already exist).

REPLACE
(input, CHAR, 7) means that if the file exists, it will be replaced with only the added records. If a specified SFS file does not exist, a new file is created. This intent is not allowed for BFS if the file does not already exist.

WRITE
(input, CHAR, 5) means that the file may be written to or read from. All changed and added records are written to the file. Other records remain unchanged. If the file does not exist, a new file is created. This intent is not allowed for BFS.

CACHE
(input, CHAR, 5) should be specified when the caller intends to read the data sequentially most of the time. Specifying this parameter causes the file system to cache several data blocks for the file, performing I/O only when the cache buffer is full (for writing) or empty (for reading). This generally reduces the number of separate I/O operations performed on the file.

NOCACHE
(input, CHAR, 7) should be specified when the caller intends to read the data in random order most of the time (not sequentially).

F
(input, CHAR, 1) means fixed (F) length records are being used. This is the default. The format attribute (F or V) is used only when the open intent is NEW, REPLACE, or WRITE (for a nonexistent file).

V
(input, CHAR, 1) means variable (V) length records are being used. The format attribute (F or V) is used only when the open intent is NEW, REPLACE, or WRITE (for a nonexistent file). If the format is not specified, the default is F. The V attribute is not allowed for BFS.

NEWDATEREF
(input, CHAR, 10) specifies that SFS should update the date of last reference for the file for any open intent. The date of last reference is the date the file was last read or modified. NEWDATEREF is the default. This parameter is not applicable for a minidisk file.

OLDDATEREF
(input, CHAR, 10) specifies that SFS should not change the date of last reference when the file is opened with an intent of READ. The date of last reference is the date the file was last read or modified. OLDDATEREF is ignored when the file is opened with intents other than READ, because they allow the file to be changed. This parameter is not applicable for a minidisk file.

RECOVER
(input, CHAR, 7) specifies that all uncommitted changes are discarded when a work unit is rolled back. The rollback can be initiated by the application, or it can be caused by abend processing or system failure.

The recoverability attribute (RECOVER or NORECOVER) is assigned to the file only when the open intent is NEW, REPLACE, or WRITE (for a nonexistent file). If the recoverability attribute is not

specified for one of these intents, a value is assigned. See usage notes “19” on page 347 and “20” on page 347. If an overwrite attribute of INPLACE is specified or assigned, the default recoverability attribute is NORECOVER.

All minidisk files are nonrecoverable. A warning will result if the RECOVER parameter is specified for a minidisk file.

NORECOVER
(input, CHAR, 9) specifies that changes to the file are not rolled back when the application initiates a rollback. In most cases, the updates are committed.

In implicit rollbacks, changes are committed if possible; in particular, cached updates can be discarded.

The recoverability attribute (RECOVER or NORECOVER) is assigned to the file only when the open intent is NEW, REPLACE, or WRITE (for a nonexistent file). If the recoverability attribute is not specified for one of these intents, a value is assigned. See usage notes “19” on page 347 and “20” on page 347. If an overwrite attribute of INPLACE is specified or assigned, the default recoverability attribute is NORECOVER.

This parameter is not applicable for a minidisk file. All minidisk files are nonrecoverable.

This attribute is not allowed for BFS.

NOTINPLACE
(input, CHAR, 10) specifies that readers see a consistent version of the file from Open to Close.

The overwrite attribute (NOTINPLACE or INPLACE) is assigned to the file only when the open intent is NEW, REPLACE, or WRITE (for a nonexistent file). If the overwrite attribute is not specified for one of these intents, a value is assigned. See usage notes “19” on page 347 and “20” on page 347.

This parameter is not applicable for a minidisk file. Minidisk file mode numbers from 0-5 are treated as NOTINPLACE. A warning will result if NOTINPLACE is specified for a minidisk file when the file mode number is 6.

INPLACE
(input, CHAR, 7) specifies that updates are to be made in place where possible. This can reduce DASD utilization and enables readers to see changes made by a concurrent writer without closing and reopening the file. Users that have an INPLACE file open for reading have to reopen the file to see extensions (new records or blocks) that have been written and committed to the file.

The overwrite attribute (NOTINPLACE or INPLACE) is assigned to the file only when the open intent is NEW, REPLACE, or WRITE (for a nonexistent file). If the overwrite attribute is not specified for one of these intents, a value is assigned. See usage notes “19” on page 347 and “20” on page 347.

DMSOPEN REPLACE replaces an existing file or creates a new one. Write operations performed on a file that has been opened for REPLACE are always shadowed until the first changes are committed, even if the file is defined with the INPLACE attribute.

The INPLACE parameter is not applicable for a minidisk file. Minidisk file mode numbers from 0-5 are treated as NOTINPLACE. A warning results if INPLACE is specified for a minidisk file when file mode number is not 6.

This attribute is not allowed for BFS.

OPENRECOVER
(input, CHAR, 11) indicates that all updates to the file resulting from this Open will be treated as if the file’s recoverability and overwrite attributes were RECOVER and NOTINPLACE. Specifying this parameter does not affect the assignment of recoverability and overwrite attributes to the file. Once the file is closed and committed, the changes to the file and any subsequent activity are handled in a manner consistent with the attributes assigned to the file.

OPENRECOVER is appropriate for pruning a log file. Usually you want such a file to be INPLACE and NORECOVER, so that each change is committed when it is made. However, when you are pruning the file you want all or none of the changes to be made.

This parameter is not applicable for a minidisk file.

ALLOWEMPTY
(input, CHAR, 10) indicates that a file with no records can be created if there are no records in the file at the time a DMSCLOSE or COMMIT is issued: either a new empty file is created or an existing file is replaced with an empty file.

This parameter is not applicable for a minidisk file. Empty files are not allowed on minidisks. A warning will result if ALLOWEMPTY is specified for a minidisk file.

SHORTDATE
(input, CHAR, 9) indicates the format of the *create_date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *create_date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *create_date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (NEW, READ, REPLACE, or WRITE; plus CACHE or NOCACHE, if specified; plus F or V, if specified; plus NEWDATEREF or OLDDATEREF, if specified; plus RECOVER or NORECOVER, if specified; plus NOTINPLACE or INPLACE, if specified; plus OPENRECOVER and ALLOWEMPTY, if specified; plus SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

token
(output, CHAR, 8) is a variable for returning a value that uniquely identifies the file being opened. Specify the token on subsequent Read (DMSREAD), Write (DMSWRITE), and Close (DMSCLOSE) routines related to this Open.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the work unit ID parameter, specify a value of 0 for *workunitid*.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see the usage note "14" on page 346).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

create_date
(input, CHAR, 8 or 10) is the Coordinated Universal Time (UTC) date-of-creation.

This parameter is applicable only if the file is a new SFS file. (That is, the file did not exist prior to the execution of DMSOPEN.)

It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

If the file is new, you can have the system determine the creation date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

create_time
(input, CHAR, 8) is the Coordinated Universal Time (UTC) time-of-creation.

This parameter is applicable only if the file is a new SFS file. That is, the file did not exist prior to the execution of DMSOPEN.

If the file is new, you can have the system determine the creation time by omitting the parameter or by specifying 8 blanks.

Use a colon (:) as the separator character and 2 digits for each position in the form *hh:mm:ss*. For example,

```
15:09:17
```

lrecl
(input, INT, 4) is an optional parameter for assigning the record length for fixed-length (F) format when a file is opened with an intent of NEW, WRITE, or REPLACE. If this parameter is omitted or 0 is specified when a file is opened that does not exist or that is being replaced, the record length of the file is determined by the system when the file is first written to.

A warning reason code is issued when an existing fixed-length file is opened and *lrecl* does not match that of the file.

If the specified value of *lrecl* does not match the record length determined by the system, an return code is issued when the file is first written to.

For a BFS file, only a value of 1 is allowed.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid* for use on later Check (DMSCHECK) requests.

The processing of a given DMSOPEN request may not require communication with the file pool server. In this case, the operation is performed synchronously regardless of the value specified in *requestid* and the value of the *requestid* parameter is not changed.

Usage Notes

1. If a file is opened for REPLACE, reading before writing gives an end-of-file return code. Only records written after opening can be read.

2. If an SFS or BFS file is opened for REPLACE and then closed before any records are written, the file remains unchanged unless ALLOWEMPTY is also specified. Minidisk files, when opened for REPLACE, are erased when the file is opened. Closing the minidisk file without writing to it has the effect of erasing the file.
3. If an SFS or minidisk file is opened for REPLACE or WRITE and the file does not exist, the open is accepted. If the record format (F or V) is not specified, it defaults to fixed (F).
4. If a file that does not exist is opened (REPLACE, WRITE, or NEW), it is not visible to other applications until records are committed.
5. There is no default for the CACHE|NOCACHE option. If neither is specified, the system chooses the most appropriate method.
6. Committing changes makes them available to readers, even if you keep the SFS or BFS file open.
7. DMSOPEN will allow a CMS file to be opened more than once by an application. Because SFS offers better sharing and data integrity than minidisks, the following conditions apply:
   - An SFS or BFS file can be opened more than once for reading *and* once for output (NEW, WRITE or REPLACE).
   - A minidisk file can be opened more than once for reading *or* once for output. Once the file has been opened for output, it must be closed before it can be reopened.
8. Once a file is open, you can use the Extract/Replace routine to retrieve the Active File Set (AFS), which contains certain data that are not available when the file is not open.

   See DMSERP for more information on Extract/Replace.
9. You will receive an error if you attempt to use the Open routine with the READ option on a file that does not exist.
10. If you choose the NEW, REPLACE, or WRITE option and a file or directory is locked, the open will fail. An exception is when the issuer has an UPDATE or EXCLUSIVE lock on the file or directory.
11. If you choose the READ option and a file or directory is locked, the open is allowed. An exception is when another user has an EXCLUSIVE lock on the file or directory. In this case the open is not allowed.
12. When reading files that reside in directory control directories, remember that:
    - You do not see changes made to a directory during a period that you have it accessed read-only. To see changes, you must reaccess the directory.
    - If you do not have the directory accessed, you must close and reopen the file to see any changes made to the file after you opened it.

    Exception: You can see changes to INPLACE files without reopening the file or reaccessing the directory.
13. You cannot open a file with the NEW, REPLACE, or WRITE option if:
    - Another user has the file or directory locked
    - The file resides in a directory control directory, and another user has the directory accessed R/W
    - The file resides in a directory control directory, and you have the directory accessed R/O
    - (Applies only to REPLACE and WRITE options) the file is an alias residing in a file control directory accessed R/W, and its base file resides in a directory control directory that is accessed read-only.
    - The file resides on a minidisk and is already opened for output using the FSOPEN interface.
14. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file you are opening. If the file being opened is on a minidisk, *userdata* is not used.
15. If you specify OLDDATEREF or NEWDATEREF and execute the routine against a file pool that does not support dates of last reference, the parameters are ignored.

16. File pool servers that are not at the current release level may not support the creation date and creation time attributes. Those servers ignore *create_date* or *create_time*.
17. In general, you cannot see another user's uncommitted changes, but if a file opened for reading has the INPLACE attribute, this is not true.

    To make viewing of concurrent updates easier for the reader of an INPLACE file, the reader should use the REFRESH option of DMSREAD, and the writer should use the FORCE option of DMSWRITE. See DMSREAD and DMSWRITE for details.
18. You can see your own uncommitted changes on the same work unit. If you update and close a file without committing the changes, and then reopen it on the same work unit, you will see the updated version. If you do not close the file and open it a second time, or if you open it on another work unit, you only see the last committed version of the file.
19. If you do not specify recoverability and overwrite attributes for a file that is being created by this DMSOPEN, then the attributes given to the file will be based on the defaults for files with that file mode number as established by the DMSPUSHA (Push Attributes) routine. If a Push Attributes has not been done, then the recoverability attribute defaults to RECOVER, and the overwrite attribute defaults to NOTINPLACE.
20. If the file mode number, recoverability attribute, or overwrite attribute is specified for a DMSOPEN REPLACE, the resulting file uses the new attribute as specified. Otherwise, the attribute is inherited from the file being replaced, if one exists. The defaults set by DMSPUSHA (Push Attributes) do not apply when replacing an existing file.
21. When the WRITE or READ option is used on an existing file, the recoverability and overwrite specifications are not used to set the attributes of the file. However, if they are specified and they do not match those already set, a warning is issued.
22. The specification of a logical record length (*lrecl*) with the ALLOWEMPTY parameter can be used to assign a specific record length to an empty file. The record length of the file is determined by these conditions:
    - If the file is in variable-length format, the record length is set to 1.
    - If the file is in fixed-length format and *lrecl* is nonzero, *lrecl* becomes the record length of the file.
    - If the file is an existing file in fixed-length format, and *lrecl* is not specified when the file is opened for replacement, the *lrecl* will be inherited from the previous version of the file.
    - If the file is a new file in fixed-length format, and *lrecl* was not specified on a DMSOPEN, the record length defaults to 80.
23. If an existing file is in DFSMS/VM migrated status and is opened with intents of Read or Write, the file is recalled unless RECALL has been set to OFF. Opening a file for Replace does not trigger a recall.
24. If a file is in DFSMS/VM migrated status, an asynchronous DMSOPEN request with an intent of READ or WRITE causes the file to be recalled. Even if the work unit is rolled back before DMSOPEN processing finishes, the file is recalled.
25. When *wuerror* is specified on DMSOPEN and an SFS error occurs during the recall process, FPERROR is updated to include the specific reason code of the failing request and the file pool ID of the failing resource.
26. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
27. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
28. All minidisk requests are done synchronously.
29. BFS files may contain more than 2^{31}-1 records (bytes) when created by means other than CMS record interfaces. When opening a BFS file with CMS record interfaces, the open attempt will fail if the file has more than 2^{31}-1 records. An attempt to write more than 2^{31}-1 records will also fail.
30. Opening a BFS file using this interface requires file pool administration authority.

31. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *create_date* parameter.
32. When *create_date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).
33. When *create_date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
34. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSOPEN, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSOPEN. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSOPEN can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	10220	File mode value other than 1 specified for a BFS file.
WARNING	44030	Intent was WRITE or REPLACE, and the file did not previously exist.
WARNING	44035	Intent was WRITE or REPLACE, and the file did not previously exist. In addition, ALLOWEMPTY was specified but the file pool or minidisk does not support this option.
WARNING	51051	You specified nonblank values for *create_date* or *create_time* and opened an existing file with an intent of WRITE or REPLACE. You cannot change the creation date or creation time of an existing file.
WARNING	78107	Recoverability attribute does not match that of the base file. The NORECOVER attribute is not allowed for BFS.
WARNING	78108	Overwrite attribute does not match that of the base file. The INPLACE attribute is not allowed for BFS.
WARNING	90306	The specified *lrecl* did not match the logical record length of the existing fixed-length format file.
WARNING	90620	RECOVER parameter was specified for a minidisk file. All minidisk files are nonrecoverable.

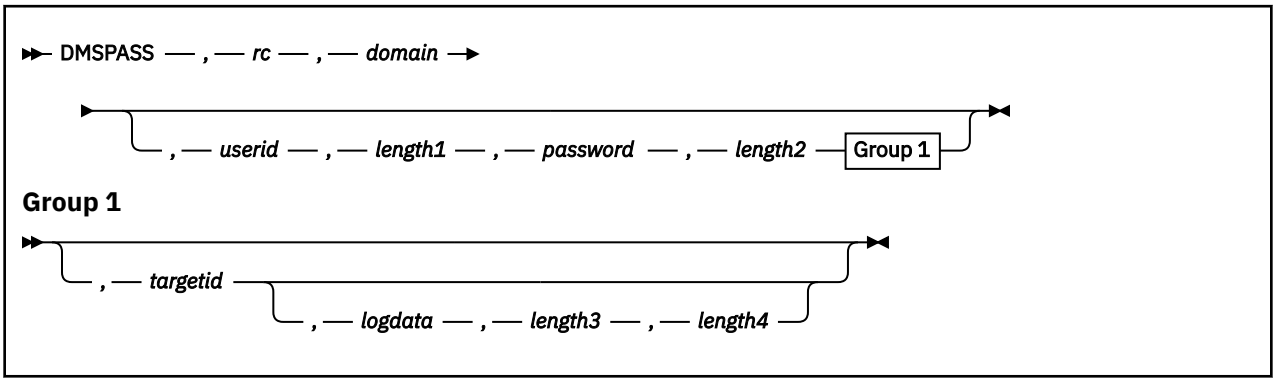
Severity	Reason Code	Description
WARNING	90621	INPLACE parameter was specified for a minidisk file with a file mode number other than 6 or NOTINPLACE was specified for a minidisk file with a file mode number of 6.
WARNING	90622	ALLOWEMPTY parameter was specified for a minidisk file.
WARNING	98700	Server managing this file does not support certain parameters of DMSOPEN: **NORECOVER** RECOVER is assumed. **INPLACE** NOTINPLACE is assumed. **ALLOWEMPTY** The server ignores the parameter. ***create_date*** The server ignores the parameter. ***create_time*** The server ignores the parameter.
ERROR	20000	Intent was NEW, but the file already exists.
ERROR	31000	Mixing recoverable and nonrecoverable work for the same file is incorrect within a single work unit.
ERROR	30500	The combination of attributes INPLACE and RECOVER is not supported.
ERROR	44000	• Intent was NEW, and the directory does not exist or you are not authorized to create a file in the directory, **or**, • Intent was WRITE or REPLACE, and the file does not exist and you are not authorized to create a file in the directory, or you are not authorized to write to the file, or the directory does not exist, **or**, • Intent was READ, and the file does not exist or you are not authorized to read the file.
ERROR	44200	Your attempt to open a file failed because you already have it open with DMSOPEN, DMSOPBLK, or DMSOPDBK: • You tried to open an SFS or minidisk file for NEW, WRITE, or REPLACE and you had already opened it for NEW, WRITE, or REPLACE. • Or you tried to open a minidisk file for WRITE or REPLACE and you had already opened it for READ. • Or you tried to open a minidisk file for READ and you had already opened it for NEW, WRITE, or REPLACE. This reason code is returned for SFS files only if both attempts to open the file occurred on the same work unit.
ERROR	50500	Attempt to exceed the maximum number of 4KB file space blocks allowed for this user. Applicable only when NEW, WRITE, or REPLACE parameter is specified.
ERROR	50600	File cannot be opened. The user has no storage blocks allocated to his file space.

Severity	Reason Code	Description
ERROR	63700	Directory control directory is accessed read-only.
ERROR	64300	The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.
ERROR	65200	File is in DFSMS/VM migrated status and implicit RECALL is set to OFF. (Set RECALL ON or issue a DFSMS RECALL command.)
ERROR	65400	Specified operation cannot be performed on an external object. The operation cannot be performed against a BFS object due to one or more of the following: • Record format other than F specified for a BFS file. • Intent was not READ or REPLACE for a BFS file. • BFS file has more than 2^{31}-1 records (bytes). • Intent was REPLACE for a BFS file and the file does not exist.
ERROR	65500	File is in DFSMS/VM migrated status and there is no active DFSMS. (The file pool server is running with the NODFSMS startup parameter in effect.)
ERROR	65900	System error. A server error occurred during an attempt to open a file for a purpose other than reading, that is, with NEW, WRITE, or REPLACE.
ERROR	66100	File is in DFSMS/VM migrated status and DFSMS/VM recall processing has been disabled.
ERROR	81058	System error. An addressing exception occurred while accessing data in a data space owned by a file pool server machine.
ERROR	90105	Incorrect record format.
ERROR	90250	The file name and file type (or *namedef*) are required but were not specified.
ERROR	90300	Incorrect intent specified. Valid intents are NEW, READ, WRITE, or REPLACE.
ERROR	90307	*lrecl* must not be negative.
ERROR	90310	Incorrect keyword specified. Valid keywords are NEW or READ or REPLACE or WRITE, CACHE or NOCACHE, F or V, OLDDATEREF or NEWDATEREF, RECOVER or NORECOVER, INPLACE or NOTINPLACE, OPENRECOVER, ALLOWEMPTY, SHORTDATE or FULLDATE or ISODATE.
ERROR	90320	Conflicting options in CSL parameter list. NEW, READ, REPLACE, and WRITE are mutually exclusive parameters, CACHE and NOCACHE, if specified, are mutually exclusive parameters, F and V, if specified, are mutually exclusive parameters, NEWDATEREF and OLDDATEREF, if specified, are mutually exclusive parameters, RECOVER and NORECOVER, if specified, are mutually exclusive parameters, INPLACE and NOTINPLACE, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Severity	Reason Code	Description
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: NEW, READ, REPLACE, WRITE, CACHE, NOCACHE, F, V, NEWDATEREF, OLDDATEREF, RECOVER, NORECOVER, INPLACE, NOTINPLACE, OPENRECOVER, ALLOWEMPTY, SHORTDATE, FULLDATE, and ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least one and not more than four tokens in the string.
ERROR	90410	Incorrect length specified for one of the character variables.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fn_ft* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fn_ft* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90440	The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.
ERROR	90450	Incorrect characters (* or %) were found in either the file name or file type part of the *fn_ft* parameter.
ERROR	90472	*requestid* must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.
ERROR	90496	The date specified is incorrect; it must be a number 0-9.
ERROR	90498	The time specified is in incorrect format; it must be in the format *hh:mm:ss*.
ERROR	90499	The time specified is incorrect; it must be a number 0-9.
ERROR	90500	The specified directory name is incorrect.
ERROR	90505	The specified directory name is of a form that represents an extended form of directory ID. The directory ID had a form such as '+A'. Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fn_ft* or *dirname* parameter is longer than 16 characters.

Severity	Reason Code	Description
ERROR	90530	The file ID of the file to be opened is incomplete or incorrect. Some possible errors are: • The namedef part of the file ID compound parameter (*fn_ft* and *dirname* or *filemode*) does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a *fn_ft* namedef was expected. • The file ID compound parameter does not resolve to a complete file ID; for example, the file mode was omitted.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.
ERROR	90602	Incorrect file mode specified. Asterisk is only allowed for open intent of READ.
ERROR	90603	Attempted to open the file for output and file mode is read/only.
ERROR	90604	You have already opened the file with the FS macro interface: If it is a minidisk file, it is open for input or output; if it is an SFS file, it is opened for output.
ERROR	90606	I/O error found when processing a minidisk file.
ERROR	90680	I/O error accessing OS dataset.
ERROR	90681	OS read password protected dataset.
ERROR	90682	OS dataset organization is not BSAM, QSAM or BPAM.
ERROR	90683	OS dataset more than 16 extents.
ERROR	90684	Attempt to open a file on an OS or DOS formatted minidisk.
ERROR	90685	Received an unexpected return code while opening a minidisk file.
ERROR	96300	Error while trying to release virtual storage in a user's machine. Further attempts to access the file pool will be rejected.

DMSPASS - Verify Logon Password and Password Phrase



Call Format

The format for calling a CSL routine is language dependent. DMSPASS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPASS
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

DMSPASS is used to verify that a particular user ID and password (or password phrase) are valid and, optionally, to verify that a user is permitted to "LOGON BY" to another user. The DMSPASS routine provides the same functionality as "DMSPWCHK - Verify Logon Password" on page 374, except it will first attempt to use diagnose code X'88' subcode 8 for all password and password phrase validation.

If CP signals the presence of an external security manager (ESM), but there is no support for password phrases using diagnose code X'88', then DMSPASS will call DMSPWCHK if the password contains 8 or fewer characters.

Parameters

rc
(output, INT, 4) is a variable for the return code from DMSPASS.

domain
(input, INT, 4) is a variable for identifying the scope of the user identifier used for authentication. Possible values are:

-1
verifies authorization to issue the CSL routine (DIAG 088).

0
identifies *userid* as a z/VM user ID.

All other values are reserved for future use.

userid
(input, CHAR, 8) is a variable for specifying the z/VM user ID to be checked.

length1
(input, INT, 4) is a variable for specifying the length of the *userid* whose password is to be verified. The maximum value is 8.

password
(input, CHAR, *) is a variable for specifying the z/VM password or password phrase for *userid*. Do not enclose a password phrase in quotation marks or double any single quotation marks.

length2
(input, INT, 4) is a variable for specifying the length of the *password*. The maximum value is 200. The External Security Manager (ESM) may restrict the length to fewer than 200 characters.

targetid
(inout, CHAR, 8) is a variable for specifying another user ID. DMSPASS will verify that *userid* has LOGON BY privileges to *targetid*. If *targetid* is not specified, or is specified as null or blank, the LOGON BY check will not be performed and DMSPASS will replace the value with *userid*.

logdata
(output, CHAR, *length1*) is a variable that will contain the details of any error condition. The text may contain information on diagnose code X'88', RPIVAL, or RACROUTE.

length3
(input, INT, 4) is a variable for specifying the length of the *logdata* field. A value of 0 indicates that no text is to be returned by DMSPASS. The maximum value is 256.

length4
(output, INT, 4) is a variable in which DMSPASS places the length of the text that was written to the *logdata* field. A value of 0 indicates that no text was written and *logdata* is unchanged.

Usage Notes

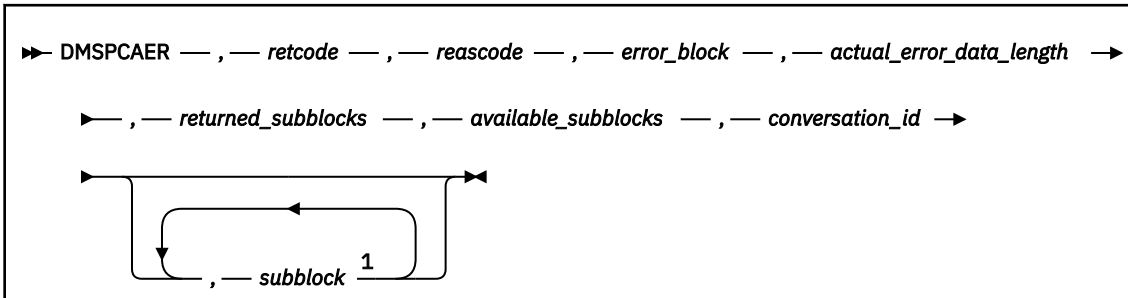
1. To use DMSPASS, the virtual machine must have OPTION DIAG88 specified in its directory entry. If an ESM is being used, additional or alternate authorization may be required. For more information on diagnose code X'88', see *z/VM: CP Programming Services*.
2. For information on “LOGON BY”, see the LOGON command in the *z/VM: CP Commands and Utilities Reference* or online help and the LOGONBY directory control statement in the *z/VM: CP Planning and Administration*.
3. For information about RACROUTE, see the *z/VM: Security Server RACROUTE Macro Reference*. Information about the RPIVAL command can be found in *z/VM: RACF Security Server Macros and Interfaces*.

Return Codes and Reason Codes

The following table lists the DMSPASS return codes.

Return Code	Description
0	*userid* is authenticated and is authorized to act as target.
4	*password* is correct but has expired (ESM only).
8	*userid* or *password* is not valid.
12	*userid* is authenticated but it does not have LOGON BY privileges to *targetid*.
24	I/O error while reading the CP directory.
28	No decision could be made (ESM only). Either the ESM is not available or there was an error issuing the RPIVAL command. Examine the *logdata* to determine the exact error.
32	Virtual machine not authorized to issue diagnose code X'88'.
36	Diagnose code X'88' not available
-1*nn*	Parameter *nn* is not valid.

DMSPCAER - Protected Conversation Adapter Errors



Notes:

[1] Up to 8 subblocks can be specified.

Context

Error Checking and Debugging

Call Format

The format for calling a CSL routine is language dependent. DMSPCAER is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPCAER
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPCAER routine to parse error information returned by the CMS Protected Conversation Adapter (PCA) during synchronization point processing.

Before calling DMSPCAER, an application must issue either the DMSGETSP (Get Synchronization Point Errors) routine or the DMSGETER (Get My Errors) routine to retrieve the error block and the actual length of the error data it contains. Because DMSGETSP retrieves all error blocks relating to the last sync point, your program must identify the error blocks containing the information you need by matching the resource component ID and exit ID of the CMS PCA. To get an error block with DMSGETER, specify the exit ID and the resource component ID of the CMS PCA on the routine call. For the CMS PCA, the exit ID is DMSPCA and the component ID is 5749DMS00.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPCAER.

reascode
(output, INT, 4) is a variable for the reason code from DMSPCAER.

error_block
(input, CHAR, 104) is a variable designating the buffer that contains the error block to be parsed.

actual_error_data_length
(input, INT, 4) is a variable used to pass the actual length of the error data in the error block. (This is the value that was returned by DMSGETSP or DMSGETER.)

returned_subblocks
(output, INT, 4) is a variable that indicates the number of error subblocks returned. Only those available subblocks that fit into the space provided are returned.

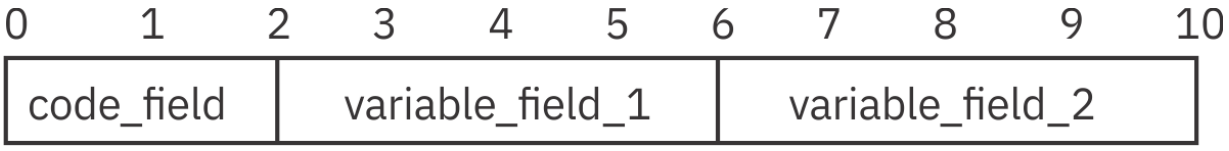
available_subblocks
(output, INT, 4) is a variable used to return the total number of error subblocks available in the error block.

conversation_id
(output, CHAR, 8) is a variable used to return the ID of the protected conversation associated with this error block.

For CPI Communications applications, this ID is the conversation ID; for assembler applications, this ID is the path ID assigned by CP (in this case, it occupies the first 2 bytes of *conversation_id*).

subblock
(output, CHAR, 10) are variables to hold parsed error subblocks. The *available_subblocks* parameter returns a value indicating how many error subblocks are available in the error block. Up to eight subblocks can be returned.

Each error subblock uniquely identifies an information, warning, or error condition that occurred during the last synchronization point. The error subblock to be returned in *subblock* has the following format:



where:

code_field
is the information, warning, or error condition.

variable_field_1
is the corresponding IPRCODE, reason code, or return code.

variable_field_2
is the corresponding action or state.

Usage Notes

1. Each call to DMSGETSP or DMSGETER returns one error block, which can then be parsed by DMSPCAER. If several error blocks are available, you should call DMSPCAER after each call to either DMSGETSP or DMSGETER that yields an error block.
2. The following tables show details about the information the CMS PCA reports in the error block. The information is categorized by the following types of conditions:
   - Information
   - Warning
   - Error
   - Full Block.

   Additional information on the *action* and *function* values of variable field 2 can be found in the ADAPTRC macro, which is described in the *z/VM: CP Planning and Administration*.

Table 35. Detailed Error Passback Information Subblock Codes

Code	Descriptive Name	Variable Field 1	Variable Field 2
X'0002'	Function_Completed_with_Heuristic_Mixed	null	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPHEURMX. APPC/VM function completing is SENDPREP, SENDCMTD, or SENDBACK.			

Table 35. Detailed Error Passback Information Subblock Codes (continued)

Code	Descriptive Name	Variable Field 1	Variable Field 2
X'0003'	Function_Completed_with_Deallocate_Abend	null	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPBACK and IPRESYNC=ON. APPC/VM function completing is SENDPREP.			
X'0004'	Backout_Resync_in_Progress	IPCODE	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPSABEND. Action is any Coordination action except Prepare_to_Resync, Ok_backout, APPC/VM function completing is SENDPREP, SENDRQCM, SENDCMTD, SENDBACK, SENDFRGT, SENDLUW, or SENDHMIX.			
X'0005'	Function_Completed_with_Deallocate_Flush	null	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPSNORM. APPC/VM function completing is SENDBACK.			
X'0006'	Function_Completed_with_Backout	null	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPBACK. APPC/VM function completing is SENDPREP or SENDRQCM.			
X'0007'	Function_Completed_with_Send_Error	null	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPERROR. APPC/VM function completing is SENDPREP.			
X'0008'	Forget_Resync_in_Progress	null	IPSENDOP
Description: During Coordination, APPC/VM function finished with IPWHATRC=IPFORGET and IPRESYNC=ON. APPC/VM function completing is SENDCMTD.			

Table 36. Detailed Error Passback Warning Subblock Codes

Code	Descriptive Name	Variable Field 1	Variable Field 2
X'4000'	Commit_State_Check	null	IPSTATE
Description: During Precoordination Commit, APPCVM QRYSTATE returned IPSTATE that is not valid.			
X'4001'	Backout_State_Check	null	IPSTATE
Description: During Precoordination Backout, APPCVM QRYSTATE returned IPSTATE that is not valid.			
X'4002'	ACCEPT_OK	null	null
Description: During Precoordination Backout, CMSIUCV ACCEPT was successful (return code <= 2 from CMSIUCV ACCEPT).			
X'4003'	IUCV_SEVER_OK	null	action
Description: During Coordination, IUCV SEVER was successful (return code <= 2 from CMSIUCV SEVER). Action is any valid Coordination action.			
X'4004'	CPI-Communications_Error_Postcoord	return code*	null
Description: During Post-Coordination, CPI-Communications return code was not 0. This can occur during any Post-Coordination action.			

*return codes:

24
Invalid parameter passed to CPI-Communications

28
CPI Communications was unable to sever this conversation.

Table 36. Detailed Error Passback Warning Subblock Codes (continued)

Code	Descriptive Name	Variable Field 1	Variable Field 2
X'4005'	Backout_Path_Severed	IPRCODE	action
Description: During Coordination, SENDBACK finishes with CC=1 and an IPRCODE that indicates the path was severed. Action = Backout, Initiator_backout, or Initiator_backout_RIP.			
X'4006'	APPCVM_Sever_OK	null	action
Description: During Coordination, APPCVM SEVER was successful (return code <=2 from CMSIUCV SEVER). Action is Deallocate_Abend.			
X'4007'	Logical_Record_Length_Error	null	IPSTATE
Description: During Precoordination Commit, APPCVM QRYSTATE returned IPSTATE = IPSENDST (SEND state) and the length of the logical record in progress was not 0.			
X'4008'	RECEIVE_Error_Info_in_IPAUDIT1	IPAUDIT	IPBFADR1
Description: During Coordination, either SENDPREP or SENDBACK finished with SENDERR with log data. Trying to RECEIVE the log data fails with CC=3 and error information in IPAUDIT1. This indicates an error in your RECEIVE buffer.			
X'4009'	RECEIVE_Error_Info_in_IPAUDIT2_or_IPAUDIT3	IPAUDIT	null or IPBFLN1F
Description: During Coordination, either SENDPREP or SENDBACK finished with SENDERR with log data. Trying to RECEIVE the log data fails with CC=3 and error information in IPAUDIT2 or IPAUDIT3. If there is error information in IPAUDIT2, then variable field 2 is null. If there is error information in IPAUDIT3, then variable field 2 is in IPBFLN1F			
X'400A'	Error_Writing_Log_Data	return code	null
Description: During Coordination, either SENDPREP or SENDBACK finished with SENDERR with log data. Attempt to write log data to CMSCOMM LOGDATA was unsuccessful.			
X'400B'	CPI-Communications_Error_Precoord	return code*	null
Description: During Pre-Coordination, CPI-Communications return code was not 0. This can occur during any Pre-Coordination action. *return codes: **10** CPI-Communications is busy and is not available for synchronization point. **24** Invalid parameter passed to CPI Communications **37** Operation was not accepted. The conversation is busy.			

Table 37. Detailed Error Passback Error Subblock Codes

Code	Descriptive Name	Variable Field 1	Variable Field 2
X'8000'	QRYSTATE_Error_Precoord	IPRCODE	action
Description: During Precoordination (Commit or Backout), APPCVM QRYSTATE finished with CC=1.			
X'8001'	HNDIUCV_SET_Error	return code	null
Description: During Precoordination Backout, HNDIUCV SET return code ¬= 0.			
X'8002'	ACCEPT_Error	CMSIUCV return code	null

Table 37. Detailed Error Passback Error Subblock Codes (continued)

Code	Descriptive Name	Variable Field 1	Variable Field 2
Description: During Precoordination Backout, CMSIUCV ACCEPT was unsuccessful (return code < 2 from CMSIUCV ACCEPT).			
X'8003'	APPCVM_Sever_Error	CMSIUCV return code	action
Description: During Coordination, APPCVM SEVER was unsuccessful (return code < 2 from CMSIUCV SEVER). Action is Deallocate_Abend.			
X'8004'	IUCV_Sever_Error	CMSIUCV return code	action
Description: During Coordination, IUCV SEVER was unsuccessful (return code < 2 from CMSIUCV SEVER). Action is any valid Coordination action.			
X'8005'	Syncpoint_Manager_Error_Precoord	SPM reason code	SPM action
Description: During Precoordination Commit, Syncpoint Manager return code ¬= 0. Contact your service representative.			
X'8006'	Syncpoint_Manager_Error_Coord	SPM reason code	SPM action
Description: During Coordination, SPM return code ¬= 0. Action = Prepare or Initiator_request_commit_new_luwid_ok. Contact your service representative.			
X'8007'	QRYSTATE_Error_Postcoord	IPRCODE	action
Description: During Post-Coordination, APPCVM QRYSTATE finished with CC=1. Action is any Post-Coordination action.			
X'8008'	Coordination_State_Check	IPRCODE	IPSENDOP
Description: During Coordination, APPC/VM function finishes with CC=1 and IPRCODE indicating a state check. APPC/VM function completing is SENDPREP, SENDRQCM, SENDCMTD, SENDFRGT, SENDLUW, SENDHMIX, SENDBACK, or SENDCNFD.			
X'8009'	Error_Before_Function_Initiated	IPRCODE	IPSENDOP
Description: During Coordination, APPC/VM function finishes with CC=1 and IPRCODE not indicating a state check. APPC/VM function completing is SENDBACK.			
X'800A'	Invalid_Function_Completion	IPWHATRC	IPSENDOP
Description: During Coordination, APPC/VM function finishes with CC=2 and invalid IPWHATRC or CC=0 and invalid IPSENDOP (on asynchronous completion). APPC/VM function completing is SENDPREP, SENDRQCM, SENDCMTD, SENDFRGT, SENDLUW, SENDHMIX, SENDBACK, or SENDCNFD.			
X'800B'	Backout_State_Failure	null	IPSTATE
Description: During Precoordination Backout, QRYSTATE returns IPSTATE = IPCOMMIT (Committed_Received state is invalid here).			
X'800C'	Error_Transferring_LUWID	IPAUDIT	IPSENDOP
Description: During Coordination, APPC/VM function finished with CC=2 and error information in IPAUDIT. APPC/VM function finishing is SENDLUW or SENDRQCM.			
X'800E'	Interrupt_Error	IPTYPE	IPSENDOP
Description: During Coordination interrupt processing, the interrupt has an invalid interrupt type. Interrupt type must be function complete.			

Table 37. Detailed Error Passback Error Subblock Codes (continued)

Code	Descriptive Name	Variable Field 1	Variable Field 2
X'800F'	CMSSTOR_Error	return code	action
Description: CMSSTOR finished unsuccessfully during Coordination. Action = New_LUWID, Initiator_request_commit_new_luwid_ok, Committed_with_new_luwid. Prepare, Backout, Initiator_backout, or Initiator_backout_RIP.			
X'8010'	Path_not_active	adapter token	function
Description: When attempting to begin sync point processing, the path was not active.			
X'8011'	Connect_State_Contradicts_Path_Table_Info	null	null
Description: During Precoordination Backout, the conversation is in CONNECT state and the path table entry indicates that the path has already been accepted.			

Table 38. Detailed Error Passback Full Block Code

Code	Description
X'FFFF'	Error_Block_full_error_info_lost
Description: Error block was full when append of subblock attempted. Subblock was not appended. Error information in subblock was lost. The user only sees this code if this condition occurs. If the block was not full when the append was attempted (for example, the append was successful), the user will not see the Full_block_code.	

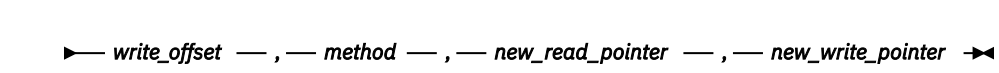
Return Codes and Reason Codes

For lists of the possible return codes from DMSPCAER, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSPCAER. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Severity	Reason Code	Description
WARNING	90271	Full error block. Some error data was lost.
WARNING	90278	No error data was available (*actual_error_data_length*=4).
ERROR	90320	Invalid *error_block* or *actual_error_data_length* parameter.

DMSPOINT - Point



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSPOINT, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPOINT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPOINT routine to move the read and write pointers of a file previously opened using the DMSOPEN (Open) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPOINT.

reascode
(output, INT, 4) is a variable for the reason code from DMSPOINT.

token
(input, CHAR, 8) is a variable for passing the value that uniquely identifies the file. This value was returned from previous call to DMSOPEN.

read_offset
(input, INT, 4) is a signed variable specifying the record to be read by the next sequential DMSREAD operation. The values can be:

0
indicates no change.

n
indicates an absolute record number or offset, depending on the value in *method*.

write_offset
(input, INT, 4) is a signed variable specifying the record to be written by the next sequential DMSWRITE operation. The values can be:

0
indicates no change.

n
indicates an absolute record number or offset, depending on the value in *method*.

method
(input, INT, 4) is a signed variable containing a code that determines how the values in *read_offset* and *write_offset* are used.

Code	Meaning of offset	Use of offset
0	Offset from the beginning of the file	An offset of -1 moves the pointer to the end of the file. An offset less than -1 is an error.
1	Signed offset from current record pointer	The offset is added to the current pointer to determine the new record pointer.
2	Offset from the end of the file	The offset is subtracted from the end-of-file pointer.

new_read_pointer
(output, INT, 4) is a signed variable containing the resulting read pointer.

new_write_pointer
(output, INT, 4) is a signed variable containing the resulting write pointer.

Usage Notes

1. Computed values of *new_write_pointer* and *new_read_pointer* must be within the range 1 to 2^{31} -1. An error results for numbers outside this range.
2. Both the read and write pointers can be changed in one call to DMSPOINT. However, both pointers must use either absolute record pointers or offsets from the current pointers.
3. The read and write offsets can be positive or negative, and can be added to the current pointer (*method*=1) or subtracted from the end-of-file pointer (*method*=2). It is permissible to set the pointer beyond the end of the file, but it is an error if the pointer is set before the beginning of the file.
4. To set the pointer to the end of the file (past the last record), make the offset -1 and the method 0.
5. To determine the current record pointers, set the read and write offsets to 0 and the method to 1.
6. The *new_write_pointer* and *new_read_pointer* can be outside the range of the file. An example of this would be to set the write pointer 100 records past the end of the file. Zero and negative values result in an error.
7. The write pointer may be manipulated even if the file was opened with the intent of READ.

Return Codes and Reason Codes

For lists of the possible return codes from DMSPOINT, see Appendix D, “Return Codes,” on page 597.

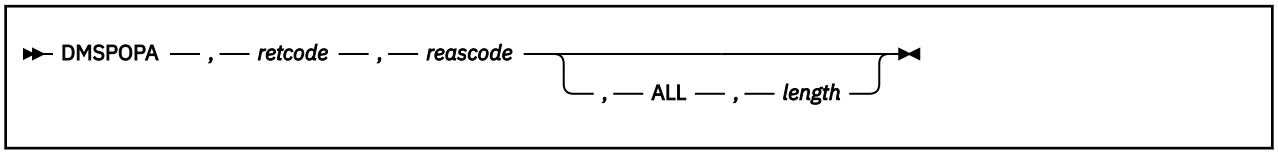
The following table lists the special reason codes returned by DMSPOINT. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSPOINT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90640	Incorrect method. Value must be 0, 1, or 2.
ERROR	90641	Incorrect *new_read_offset*. Resulting read pointer must be between 1 and 2^{31}-1.
ERROR	90642	Incorrect *new_write_offset*. Resulting write pointer must be between 1 and 2^{31}-1.

Severity	Reason Code	Description
ERROR	95750	Incorrect token specified. File not open or not opened using DMSOPEN.

DMSPOPA - Pop Attribute



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSPOPA is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPOPA
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSPOPA" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPOPA routine to pop (or remove) any recoverability and overwrite default attributes specified previously with the DMSPUSHA (Push Attributes) routine.

The set of attributes most recently pushed onto the stack will be removed and the previous set of attributes will become the active defaults.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPOPA.

reascode
(output, INT, 4) is a variable for the reason code from DMSPOPA.

ALL
(input, CHAR, 3) indicates that all sets of attribute defaults will be removed from the stack except for the CMS default of RECOVER NOTINPLACE. If in CMS Subset mode, only those defaults acquired in CMS Subset mode will be removed.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (ALL).

Return Codes and Reason Codes

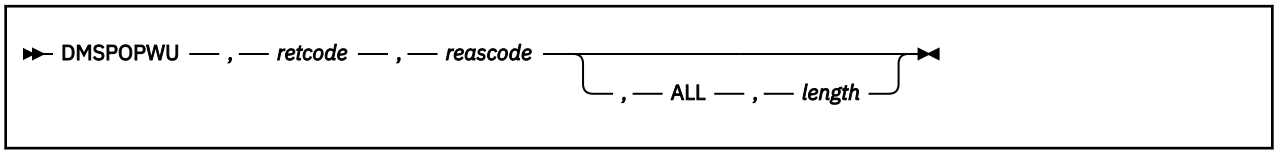
For lists of the possible return codes from DMSPOPA, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSPOPA. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSPOPA can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	78104	No file attributes were placed on the stack.
ERROR	90302	Incorrect input parameter in CSL parameter list. The parameter was not ALL or the parameter length was incorrect.

DMSPOPWU - Pop Default Work Unit ID



Context

Work Unit Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSPOPWU is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPOPWU
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPOPWU routine to remove a work unit ID from the work unit ID stack. The top default work unit ID in the stack is removed, and the previous default becomes the active default.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPOPWU.

reascode
(output, INT, 4) is a variable for the reason code from DMSPOPWU.

ALL
(input, CHAR, 3) indicates that all work unit IDs are to be removed, except for the CMS default. If in CMS subset mode, only those work unit IDs obtained while in CMS subset mode are removed.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (ALL).

Usage Notes

1. When popping individual work unit IDs, they may be popped down to, but not including, the system default. If in CMS mode, this is work unit ID 1. If in subset mode, this is the subset work unit ID.
2. Specify ALL with care. You may pop a work unit ID that another subprogram placed on the work unit ID stack and plans to use again.
3. Push Default Work Unit ID (DMSPUSWU) adds a work unit ID to the stack.

Return Codes and Reason Codes

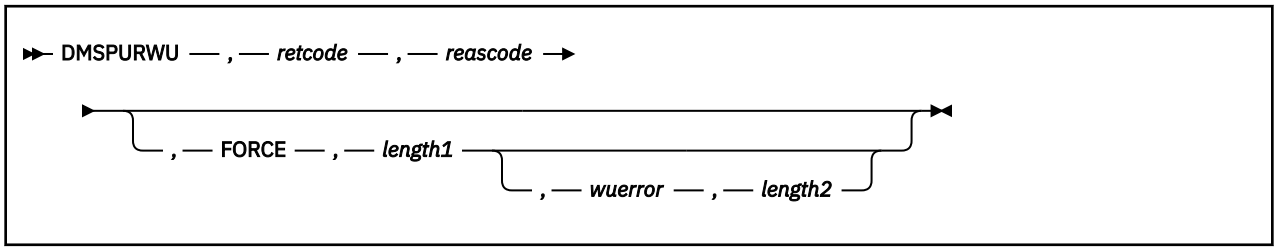
For lists of the possible return codes from DMSPOPWU, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason code returned by DMSPOPWU. ERROR means the request failed. Errors cause return code 8 or 12.

DMSPOPWU can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90300	Invalid input parameter in CSL parameter list. The parameter was not ALL.

DMSPURWU - Purge Work Unit IDs



Context

Work Unit Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSPURWU is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPURWU
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPURWU routine to return all work unit IDs to CMS and end work unit communication.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPURWU.

reascode
(output, INT, 4) is a variable for the reason code from DMSPURWU.

FORCE
(input, CHAR, 5) indicates that the work unit IDs are to be purged even if a work unit is active. If there is an active work unit, a rollback occurs. If FORCE is not specified, an active work unit causes the request to fail with no work unit IDs purged. To omit this parameter while specifying the *wuerror* parameter, specify a string of blanks as the value of FORCE.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (FORCE).

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

Usage Notes

1. Upon successful completion of DMSPURWU, communication to all file pools ends and all session locks in those file pools for this user are released. All protected conversations are deallocated at this time. Any uncommitted work is rolled back before deallocating protected conversations.
2. If another subprogram is actively using any work unit ID, an error occurs.
3. Work units that were associated with a user ID, security token, or accounting user data when the work unit IDs were assigned by DMSGETWU are no longer associated with these items after the work unit has been purged.
4. The work unit stack is not checked or modified by this routine.

Return Codes and Reason Codes

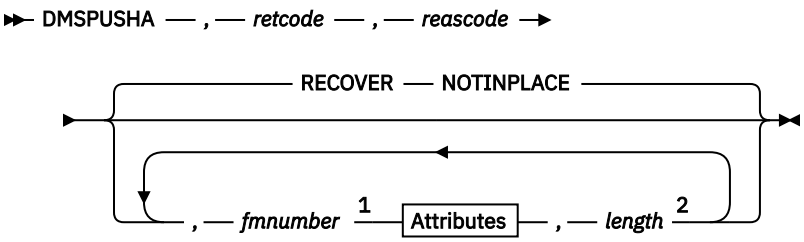
For lists of the possible return codes from DMSPURWU, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSPURWU. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

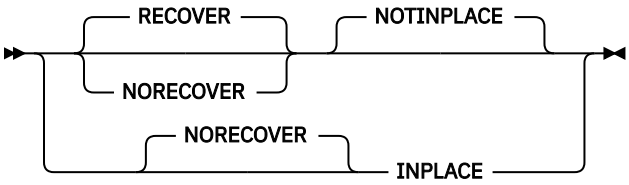
DMSPURWU can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	95950	Logical unit of work was in process for an active work unit ID and FORCE option was specified. Any active work has been rolled back.
ERROR	90300	Invalid input parameter in CSL parameter list. The only valid input parameter is FORCE.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	95800	System error. Incorrect work unit range passed to SFS.
ERROR	95900	Logical unit of work in process for an active work unit ID and FORCE option not specified. No communication paths were severed.

DMSPUSHA - Push Attributes



Attributes



Notes:

1 Any file mode number not specified is reset to the default values of RECOVER and NOTINPLACE.
2 Specify attributes for 1 to 7 file mode numbers.

Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSPUSHA is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPUSHA
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPUSHA routine to define a set of default recoverability and overwrite attributes for each file mode number. These attributes will be the system defaults when creating a new SFS file with a given file mode number, provided that neither the recoverability nor overwrite attribute has been specified on the command or CSL routine that generates the file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPUSHA.

reascode
(output, INT, 4) is a variable for the reason code from DMSPUSHA.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number for which the recoverability and overwrite attributes are being defined. This must be a single character digit from '0' through '6'.

RECOVER
(input, CHAR, 7) specifies that the file is to be recoverable. When a file is recoverable, uncommitted changes are rolled back as the result of an application-initiated rollback or system failure. RECOVER is the default.

NORECOVER
(input, CHAR, 9) specifies that changes to the file are not rolled back in the event of an application-initiated rollback. In most cases, the updates will be committed.

NOTINPLACE
(input, CHAR, 10) specifies that readers see a consistent version of the file from Open to Close. NOTINPLACE is the default.

INPLACE
(input, CHAR, 7) specifies that updates are to be made in place where possible. This can reduce DASD utilization and enables readers to see file updates by a concurrent writer without closing and reopening the file. Users that have an INPLACE file open for READ or WRITE have to reopen the file to see extensions (new records or blocks) that have been written and committed to the file.

length
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter: *fmnumber*, RECOVER or NORECOVER, and NOTINPLACE or INPLACE.

Usage Notes

1. A user can specify none, some, or all possible file mode numbers using DMSPUSHA.
2. Each time you invoke DMSPUSHA, the attributes for all file mode numbers not specified default to RECOVER and NOTINPLACE. If, for example, file mode number 6 is given the attributes NORECOVER and INPLACE using DMSPUSHA, and later a DMSPUSHA specifies attributes for file mode number 2 only, the attributes for file mode number 6 and all other unspecified file mode numbers default to RECOVER and NOTINPLACE.
3. If no parameters (other than *retcode* and *reascode*) are specified, the default of RECOVER and NOTINPLACE for each file mode number will be pushed on the stack.
4. Specifying a file mode number and only the INPLACE attribute results in the NORECOVER and INPLACE attributes being pushed onto the stack for that file mode number.
5. The attributes defined by DMSPUSHA will remain the default until one of the following events occurs:
   - A subsequent DMSPUSHA supersedes the file attributes defined
   - A subsequent DMSPOPA removes these attributes from the stack
   - The user's application is terminated either normally or through ABEND recovery
   - The application either enters or exits CMS subset mode.
6. When an application enters CMS subset mode, the system starts with a new attribute stack for the purpose of isolating work undertaken in CMS subset mode. Upon return from subset mode, any attributes that were pushed onto the stack in subset mode are popped, and the default attributes in effect before entering subset mode are reactivated.
7. The recoverability and overwrite attributes of a BFS file are always RECOVER and NOTINPLACE. Attributes set with DMSPUSHA are ignored when dealing with a BFS file.

Return Codes and Reason Codes

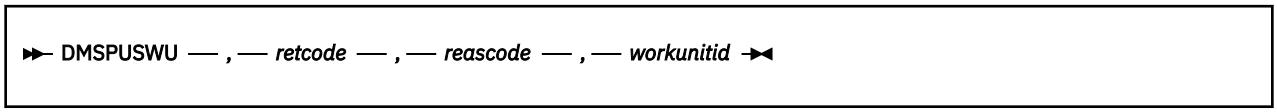
For lists of the possible return codes from DMSPUSHA, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSPUSHA. ERROR means the request failed. Errors cause return code 8 or 12.

DMSPUSHA can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	78101	Incorrect overwrite attribute specified (must be NOTINPLACE or INPLACE) or incorrect recoverability attribute specified (must be RECOVER or NORECOVER).
ERROR	78102	Duplicate attribute specification for a given file mode number.
ERROR	90300	Incorrect options in CSL parameter list.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90440	Incorrect file mode number specified.

DMSPUSWU - Push Default Work Unit ID



Context

Work Unit Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSPUSWU is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPUSWU
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPUSWU routine to make the specified work unit ID the default work unit ID for this virtual machine. This work unit ID will be used whenever a work unit ID is not specified.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSPUSWU.

reascode
(output, INT, 4) is a variable for the reason code from DMSPUSWU.

workunitid
(input, INT, 4) is a variable for specifying the work unit ID that is to be made the default. The previous default is pushed down in the default work unit ID stack.

Usage Notes

1. Multiple work unit IDs can be pushed without issuing a DMSPOPWU (Pop Default Work Unit ID).
2. Before pushing a work unit ID, you must have first obtained it by calling Get Work Unit ID (DMSGETWU).
3. The system-generated default work unit IDs cannot be pushed.

Return Codes and Reason Codes

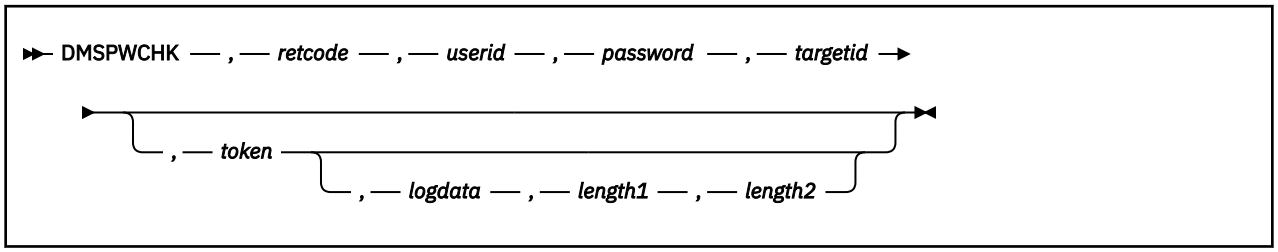
For lists of the possible return codes from DMSPUSWU, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason code returned by DMSPUSWU. ERROR means that the request failed. Errors cause return code 8 or 12.

DMSPUSWU can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	90540	Specified work unit ID is invalid.

DMSPWCHK - Verify Logon Password



Call Format

The format for calling a CSL routine is language dependent. DMSPWCHK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSPWCHK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSPWCHK routine to verify that a particular user ID and password are valid and, optionally, to verify that a user is permitted to "LOGON BY" to another user.

DMSPASS is the preferred routine for password verification. Any applications that support long passwords and password phrases must use DMSPASS.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSPWCHK.

userid
(input, CHAR, 8) is a variable for specifying the VM user ID to be checked.

password
(input, CHAR, 8) is a variable for specifying the VM password for *userid*.

targetid
(input, CHAR, 8) is a variable for specifying another user ID. DMSPWCHK will verify that *userid* has LOGON BY privileges to *targetid*. If *targetid* is specified as nulls, this check will not be performed.

token
(input, CHAR, 4) is a variable for specifying a security token. If *token* is omitted, or the value of the token is zero, then native CP security services will be used to validate the user information. If the value of the token is nonzero, standard external security manager services will be used. If the value of the token is 1 (X'00000001'), then external security services will be used, but the token will not be passed to the security service. If the value of the token is -1 (X'FFFFFFFF'), then RACROUTE will be called without an ACEE.

logdata
(output, CHAR, *length1*) is a variable that may contain arbitrary readable text passed from the CSL routine to the calling application. This text is used to more fully describe any error conditions.

Note: This text may or may not be used by the calling application.

length1
(input, INT, 4) is a variable for specifying the length of the *logdata* field. A value of zero indicates that no text is to be returned by DMSPWCHK. The maximum value is 256.

length2
(output, INT, 4) is a variable in which DMSPWCHK specifies the length of the text that it placed in *logdata*. A value of zero indicates that no text is present.

Usage Notes

1. The issuing virtual machine must have `OPTION DIAG88` specified in the CP directory entry.
2. For information on "LOGON BY", see the LOGON command in the *z/VM: CP Commands and Utilities Reference* or online help and the LOGONBY directory control statement in the *z/VM: CP Planning and Administration*.
3. DMSPWCHK communicates with the external security manager using standard interfaces, as shown in the following table:

Access control	External Security Manager Interface	Minimum Access Required
Password verification	The following CMS command is used: RPIVAL *userid password*	Application dependent
LOGON BY privilege	The RACROUTE macro is used with these parameters: REQUEST=AUTH CLASS=SURROGAT ENTITYX=LOGONBY.*targetid* USERID=*userid*	READ

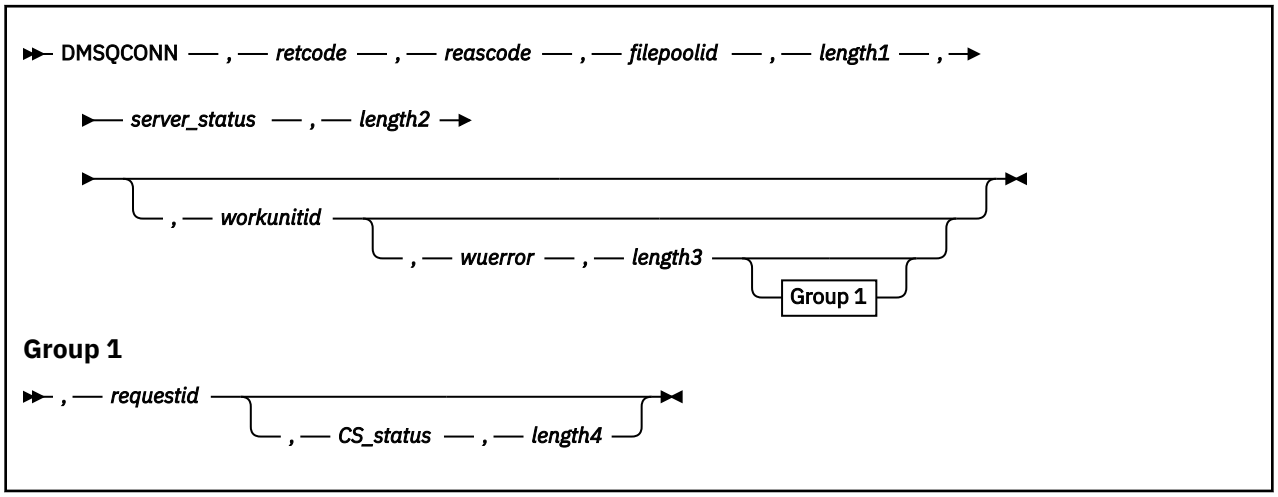
Information about RACROUTE can be found in the *z/VM: Security Server RACROUTE Macro Reference*. Information about the RPIVAL command can be found in *RACF Macros and Interfaces*.

Return Codes and Reason Codes

The following table lists the DMSPWCHK return codes.

Return Code	Description
0	Requested access is allowed.
4	Password is correct but has expired (external security manager only)
8	*userid* or *password* is not valid, or *userid* does not have LOGON BY privileges to *targetid*.
24	System paging I/O error.
28	No decision could be made (valid with external security manager only). The RPIVAL program could not be run, or the external security manager is inactive.
32	Function not authorized. See Usage Note "1" on page 375.
36	Function not available.
40	Password has expired (valid with external security manager only).
-1*nn*	Parameter *nn* is not valid.

DMSQCONN - Query Connect



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSQCONN is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQCONN
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQCONN routine to determine whether a file pool server is local (on the same VM system as the caller), located within a CS collection, or remote (connected through a communications server, such as TSAF or AVS). You must have connect authority to the file pool being queried.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQCONN.

reascode
(output, INT, 4) is a variable for the reason code from DMSQCONN.

filepoolid
(input, CHAR, 1-8) is a variable for specifying the name of the file pool (without the colon) whose location is being queried.

length1
(input, INT, 4) is a variable containing the length of the preceding character parameter (*filepoolid*).

server_status
(output, CHAR, 1) is a variable in which the routine returns a value indicating the connection status of the server:

1
indicates the server is local (on the same VM system as the caller) or located within a CS collection.

2
indicates the server is remote (connected through a communications server, such as TSAF or AVS).

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (*server_status*).

workunitid
(input, INT, 4) is a variable for identifying the work unit associated with DMSQCONN. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, this indicates that the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request.

CS_status
(output, CHAR, 1) is a variable in which the routine returns a value indicating whether the server is located within a CS collection:

0
indicates the server is not located within a CS collection.

1
indicates the server is located within a CS collection.

length4
(input, INT, 4) is a variable containing the length of the preceding character parameter (*CS_status*).

Usage Notes

1. You cannot use REXX to make an asynchronous call.
2. A request ID is returned if the request is to be asynchronous.
3. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
4. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
5. If the server is located within a CS collection, a 1 is returned in both *server_status* and *CS_status*. If a 2 is returned in *server_status*, indicating that the server is remote, *CS_status* will always be 0.

Return Codes and Reason Codes

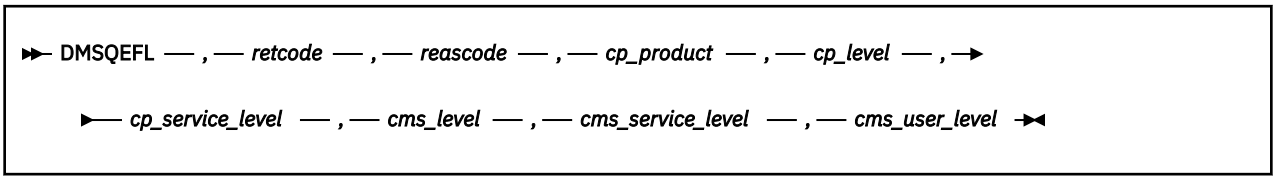
For lists of the possible return codes from DMSQCONN, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSQCONN. ERROR indicates that the request failed. Errors cause return code 8 or 12.

DMSQCONN can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid *requestid* specified; must be 0 or 1.
ERROR	90476	Invalid file pool ID specified.
ERROR	90485	Invalid length specified for server status parameter.
ERROR	90540	Specified work unit ID is invalid.

DMSQEFL - Query Functional Level of CP and CMS



Call Format

The format for calling a CSL routine is language dependent. DMSQEFL is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQEFL
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSQEFL" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQEFL routine to identify the functional level of CP and CMS.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQEFL.

reascode
(output, INT, 4) is a variable for the reason code from DMSQEFL.

cp_product
(output,INT,4) is a unique code that specifies the CP product being used. The following table shows the product codes for CP:

Table 39. CP Product Codes

Code	Product
4	VM/XA SP
8	VM/SP (including HPO)
12	VM/ESA 370 Feature
16	VM/ESA ESA
20	z/VM

Releases of CP having different product codes may have the same value for *cp_level*.

cp_level
(output,INT,4) is a unique code that specifies the functional level of CP being used. A level code can apply to releases of several different CP product. The combination of *cp_level* and *cp_product* determines the specific system environment.

The following table shows the codes for CP levels:

Table 40. CP Level Codes

Code	Level
4	VM/SP Release 5 (includes HPO) VM/XA Release 1.0 VM/ESA Version 1 Release 1.0
8	VM/SP Release 6 (includes HPO) VM/XA Release 2.0 VM/ESA Version 1 Release 1.1
12	VM/XA Release 2.1 VM/ESA Version 1 Release 2.0
16	VM/ESA Version 1 Release 2.1
20	VM/ESA Version 1 Release 2.2
24	VM/ESA Version 2 Release 1.0
28	VM/ESA Version 2 Release 2.0
32	VM/ESA Version 2 Release 3.0
36	VM/ESA Version 2 Release 4.0
40	z/VM Version 3 Release 1.0
44	z/VM Version 4 Release 1.0
48	z/VM Version 4 Release 2.0
52	z/VM Version 4 Release 3.0
56	z/VM Version 4 Release 4.0
60	z/VM Version 5 Release 1.0
64	z/VM Version 5 Release 2.0
68	z/VM Version 5 Release 3.0
72	z/VM Version 5 Release 4.0
76	z/VM Version 6 Release 1.0
80	z/VM Version 6 Release 2.0
84	z/VM Version 6 Release 3.0
88	z/VM Version 6 Release 4.0
92	z/VM Version 7 Release 1.0
96	z/VM Version 7 Release 2.0
100	z/VM Version 7 Release 3.0

Note: It is possible to get a CP level code that is not listed in Table 40 on page 380. See usage note "3" on page 382.

cp_service_level
(output,INT,4) is the CP service level in use.

cms_level
(output,INT,4) is a unique code that specifies the functional level of CMS, which is not necessarily on a release boundary. Significant between-release support items, for example, can have their own codes.

The following table shows the codes for CMS levels:

Table 41. CMS Level Codes

Code	Level	Introduced
12	CMS level 8	VM/ESA Version 1 Release 1.1
16	CMS level 9	VM/ESA Version 1 Release 2.0
20	CMS level 10	VM/ESA Version 1 Release 2.1
24	CMS level 11	VM/ESA Version 1 Release 2.2
28	CMS level 12	VM/ESA Version 2 Release 1.0
32	CMS level 13	VM/ESA Version 2 Release 2.0
36	CMS level 14	VM/ESA Version 2 Release 3.0
40	CMS level 15	VM/ESA Version 2 Release 4.0
44	CMS level 16	z/VM Version 3 Release 1.0
48	CMS level 17	z/VM Version 4 Release 1.0
52	CMS level 18	z/VM Version 4 Release 2.0
56	CMS level 19	z/VM Version 4 Release 3.0
60	CMS level 20	z/VM Version 4 Release 4.0
64	CMS level 21	z/VM Version 5 Release 1.0
68	CMS level 22	z/VM Version 5 Release 2.0
72	CMS level 23	z/VM Version 5 Release 3.0
76	CMS level 24	z/VM Version 5 Release 4.0
80	CMS level 25	z/VM Version 6 Release 1.0
84	CMS level 26	z/VM Version 6 Release 2.0
88	CMS level 27	z/VM Version 6 Release 3.0
92	CMS level 28	z/VM Version 6 Release 4.0
96	CMS level 29	z/VM Version 7 Release 1.0
100	CMS level 30	z/VM Version 7 Release 2.0
104	CMS level 31	z/VM Version 7 Release 3.0

cms_service_level
(output,INT,4) is the current CMS service level.

cms_user_level
(output,INT,4) is the current value of the USERLVL field in NUCON.

Usage Notes

1. DMSQEFL was introduced with CMS level 8 and is not available on earlier levels.
2. The 4-byte USERLVL field returned in *cms_user_level* is the same data placed in register R0 by the CMS QUERY CMSLEVEL command.

3. If this level of CMS is being used with a newer level of CP, the value returned in *cp_level* is the code for the new CP and is higher than the largest level listed in Table 40 on page 380.
4. Assembler programs can use the DMSQEFL macro to determine the CMS level. For more information, see the *z/VM: CMS Macros and Functions Reference*.

Return Codes and Reason Codes

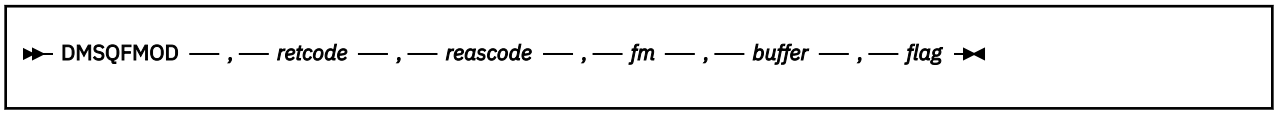
For lists of the possible return codes from DMSQEFL, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSQEFL. ERROR means the request failed. Errors cause return code 8 or 12.

DMSQEFL can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	86300	The CP product information could not be determined from the licensed program bit map contained in the CMS control blocks, possibly because of corrupted storage.
ERROR	86400	The CMS level information could not be determined, possibly because of corrupted storage.

DMSQFMOD - Query File Mode



Context

File System Management: SFS and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSQFMOD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQFMOD
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQFMOD routine to determine if a minidisk or SFS directory is accessed with the specified file mode.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQFMOD.

reascode
(output, INT, 4) is a variable for the reason code from DMSQFMOD.

fm
(input, CHAR, 1) is a variable for specifying a letter (A-Z) specifying the file mode.

buffer
(output, CHAR, 153) is a user-provided 153-byte area for returning the disk address (if minidisk) or the fully-qualified directory name. It is a character variable.

flag
(output, INT, 4) returns a value indicating whether the file mode is a directory or a minidisk and whether it is read-only or read/write. Possible values for this parameter are:

1
R/W FILECONTROL directory

2
R/O FILECONTROL directory

3
R/O minidisk

4
R/W minidisk

5
R/W DIRCONTROL directory

6
R/O DIRCONTROL directory

Usage Notes

1. The length of *buffer* must be 153 bytes. Because no validation is done to ensure that the length of *buffer* is 153 bytes, storage may be overlaid if *buffer* is not large enough to contain all the information.
2. If a disk address (for a minidisk) is returned in the buffer, it occupies the first 4 bytes of the buffer as an integer with a leading 0 if necessary.
3. When DMSQFMOD places the requested data in the buffer you provide, it does not clear any unused portion of the buffer. Therefore, it is recommended that you initialize the buffer to all blanks before each use to avoid possible confusion due to data remaining in the buffer from a previous call.

Return Codes and Reason Codes

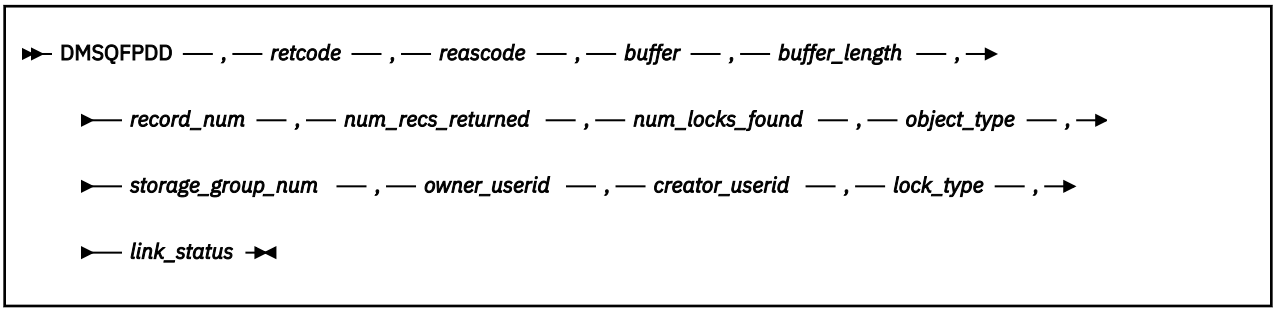
For lists of the possible return codes from DMSQFMOD, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason code returned by DMSQFMOD. ERROR indicates that the request failed. Errors cause return code 8 or 12.

DMSQFMOD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90440	The specified file mode is invalid. It must be an alphabetic character.

DMSQFPDD - Query File Pool Disable - Deblocker



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQFPDD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQFPDD
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQFPDD routine to extract one record of lock information from the buffer returned by the DMSQFPDS (Query File Pool Disable) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) contains the return code from DMSQFPDD.

reascode
(output, INT, 4) contains the reason code from DMSQFPDD.

buffer
(input, CHAR, *buffer_length*) is the variable where the lock information was stored by DMSQFPDS. One record is returned for each lock. See Table 42 on page 390 for a description of the buffer format.

buffer_length
(input, INT, 4) contains the number of bytes in *buffer*. This should be the same as the length used in DMSQFPDS.

record_num
(input, INT, 4) specifies which record in the array is to be deblocked. Specify 1 for the first record in the array.

num_recs_returned
(output, INT, 4) contains the number of records returned in the buffer array. It may be less than the number of locks found.

num_locks_found
(output, INT, 4) contains the number of locks found. If *num_locks_found* is greater than *num_recs_returned*, the buffer was not large enough to hold all the records available.

object_type
(output, CHAR, 1) contains a code indicating whether the record describes a file space or a storage group: the code is "F" for a file space and "G" for a storage group.

storage_group_num
(output, INT, 4) contains the storage group number if the record describes a storage group. Zero is returned if the issuer does not have file pool administration authority or if the record describes a file space.

owner_userid
(output, CHAR, 8) contain the user ID that owns a locked file space. Blanks are returned if the record describes a storage group.

creator_userid
(output, CHAR, 8) contains the user ID of the user or the name of the function that created the lock. At present, RENAME (for the FILEPOOL RENAME command) is the only function name that can be returned.

If the DISABLE operator command designated an owner for the lock, DMSQFPDD returns the designated owner's user ID, not the user ID that issued the command.

lock_type
(output, CHAR, 1) contains a code for the type of lock:

S
indicates a share lock.

E
indicates an exclusive lock.

B
indicates an exclusive lock created by the system.

*****
indicates the type is unknown to your CMS, because the file pool server is running at a later release level.

link_status
(output, CHAR, 1) contains a code indicating whether the minidisks in the storage group were kept linked or were detached:

D
detached

L
linked

(blank)
indicates the object is a file space or that the issuer does not have file pool administration authority.

Usage Notes

1. If *num_locks_found* is greater than *num_recs_returned*, the buffer was not large enough to hold all the information available when the Query File Pool Disable routine (DMSQFPDS) was called.
2. If *record_num* is greater than *num_recs_returned*, the record number requested is considered incorrect and an error reason code is returned.
3. If there are no errors, the deblocked information is returned.

Return Codes and Reason Codes

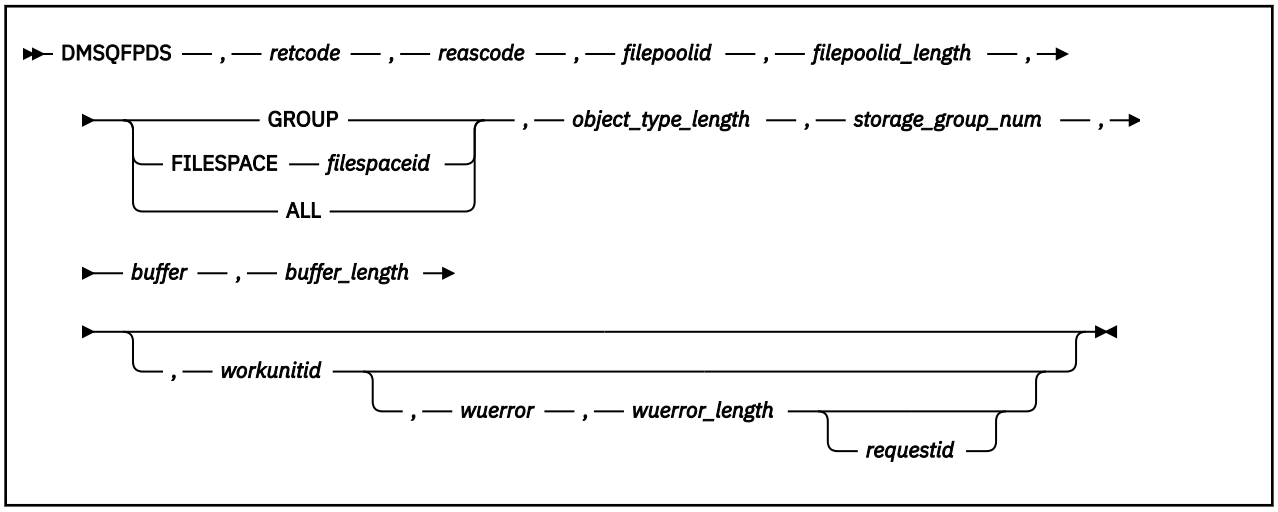
For lists of the possible return codes from DMSQFPDD, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason code returned by DMSQFPDD. ERROR means the request failed. Errors cause return code 8 or 12.

DMSQFPDD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90217	Incorrect record number specified: it is less than or equal to zero or it is greater than the number of records being returned in the buffer.

DMSQFPDS - Query File Pool Disable



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQFPDS is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQFPDS
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQFPDS routine to obtain information about locked storage groups and file spaces:

- A file pool user can get lock information about any file space in the file pool and its owning storage group.
- A file pool administrator can get information about locks on a particular storage group.
- A file pool administrator can get information about all locks on file spaces and storage groups in a file pool.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) contains the return code from DMSQFPDS.

reascode
(output, INT, 4) contains the reason code from DMSQFPDS.

filepoolid
(input, CHAR, 1-8) contains the ID of the file pool (without the colon) being queried.

filepoolid_length
(input, INT, 4) contains the length of the contents of *filepoolid*.

GROUP
(input, CHAR, 5) selects information for a storage group. File pool administration authority is necessary to use this option.

FILESPACE ***filespaceid***
(input, CHAR, 11-18) selects information for the specified file space and its storage group. The file space ID cannot be a nickname.

ALL
(input, CHAR, 3) selects information for all file spaces and all storage groups in the file pool. File pool administration authority is necessary to use this option.

object_type_length
(input, INT, 4) contains the length of the preceding character parameter (GROUP, FILESPACE *filespaceid*, or ALL).

storage_group_num
(input, INT, 4) contains the number of the storage group for which you want information. The number must be zero or between 2 and 32767, and cannot be greater than that set by the MAXDISKS control statement for the file pool. For more information about MAXDISKS, see the FILESERV GENERATE command in the *z/VM: CMS File Pool Planning, Administration, and Operation*. The storage group number must be nonzero when the object type is GROUP. If the request is not for a specific storage group, this parameter must be zero.

buffer
(output, CHAR, *buffer_length*) is a variable in which the information is returned. The buffer must be at least 40 bytes long: 12 bytes plus 28 bytes for a record of lock information. See usage note "10" on page 390 for more information about the buffer.

buffer_length
(input, INT, 4) specifies the number of bytes in the buffer.

workunitid
(input, INT, 4) contains the work unit ID for this operation. If you want to specify an optional parameter following *workunitid* without using the work unit ID parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *wuerror_length*) contains extended error information is returned. If it is specified, it must be followed by a length field.

wuerror_length
(input, INT, 4) contains the length of the preceding character parameter (*wuerror*). Specifying zero has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) contains a code indicating whether the request is to be processed asynchronously. If it is omitted or contains 0 on input, the request is synchronous. If it contains 1 on input, the request can be asynchronous. CMS returns an integer in *requestid* to identify an asynchronous request. This ID is passed on a later Check (DMSCHECK) request. CMS returns a 1 in *requestid* if the request was completed synchronously and it is not necessary to call DMSCHECK.

Usage Notes

1. Use DMSQFPDS to determine the result of these commands and routines:
   - FILEPOOL DISABLE, FILEPOOL BACKUP, FILEPOOL RESTORE, and FILEPOOL RENAME commands
   - DISABLE operator command
   - Disable File Space (DMSDISFS) and Disable Storage Group (DMSDISSG) CSL routines.

   For more information about locks on file spaces and storage groups, see the descriptions of the commands in the *z/VM: CMS File Pool Planning, Administration, and Operation*, or the descriptions of the CSL routines in this book.

2. If the DMSQFPDS routine does not resolve what type of lock exists, there may be an explicit lock on a file or directory. To get information about an explicit lock:
   - Use the DMSOPDIR (Open Directory) routine with the DMSGETDI (Get Directory) routine
   - Use the DMSGETDK (Get Directory-Lock) routine
   - Or use the QUERY LOCK command (see the *z/VM: CMS Commands and Utilities Reference*).
3. If the file space and its storage group were locked by a function, the name of the function that created the locks is returned. For example, the FILEPOOL RENAME command can put a lock on a file space and storage group. In this case RENAME is returned as the creator of the lock. (RENAME is currently the only valid function name.) For more information, see the usage notes of the FILEPOOL RENAME command in *z/VM: CMS File Pool Planning, Administration, and Operation*.
4. A return code of 0 or 4 from an asynchronous request indicates that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other request (except a rollback request) for the file pool on the same work unit.
6. DMSQFPDS is an atomic request. This means there can be no outstanding work for the file pool on the work unit when this routine is called. When it is finished, DMSQFPDS commits the work without coordination.
7. Rather than deblock the output buffer yourself, you can use the routine "DMSQFPDD - Query File Pool Disable - Deblocker" on page 385).
8. DMSQFPDS returns one record for each lock, and the buffer must be large enough to hold all the records. If more records are available than can fit in the buffer, as many records as can fit are put in the buffer, and a warning is returned. DMSQFPDS can then be reissued with the correct size for the buffer calculated from the values returned in the buffer (see Table 42 on page 390):

   ```
   (num_locks_found * length_array_rec) + 12
   ```

9. DMSQFPDS provides function similar to that of the QUERY FILEPOOL DISABLE command and the QUERY DISABLE operator command. For more information, see *z/VM: CMS File Pool Planning, Administration, and Operation*.
10. The output buffer consists of three fields followed by an array of lock information, as described in Tables Table 42 on page 390 and Table 43 on page 391.

Table 42. Format of Output Buffer

Field Name	Field Type	Description
num_recs_returned	INTEGER(4)	The number of records actually returned in the buffer array, which may be less than the number of locks found. If no locks of the specified type exist, 0 is returned in *num_recs_returned*, and a warning return code and reason code 32080 are returned.
num_locks_found	INTEGER(4)	The number of records found, which may be more than the number of records returned.
length_array_rec	INTEGER(4)	The length of each array record.
Array		One array record is returned for each lock. The format for the record is described in Table 43 on page 391.

Table 43. Format of a Record of Lock Information (Length: 28 bytes)

Field Name	Field Type	Description
object_type	CHAR(1)	A code for the type of locked object described in the record: "F" for a file space and "G" for a storage group.
	CHAR(3)	Reserved
storage_group_num	INTEGER(4)	The storage group number if the object is a storage group. Zero is returned if the object is a file space or the issuer does not have file pool administration authority.
owner_userid	CHAR(8)	If the object is a file space, the user ID of the owner or the name of the file space. Blanks are returned if the object is a storage group.
creator_userid	CHAR(8)	The ID of the user or the name of the function that created the lock (see usage note "3" on page 390 for more information on functions). If the DISABLE operator command designated an owner for the lock, DMSQFPDS returns the designated owner's user ID in this field, not the user ID that issued the command.
lock_type	CHAR(1)	A code for the type of lock: **S** indicates a share lock. **E** indicates an exclusive lock. **B** indicates an exclusive lock created by the system. ***** indicates the type is unknown to your release level of CMS, because the file pool server is running at a later release level.

Table 43. Format of a Record of Lock Information (Length: 28 bytes) (continued)		
Field Name	**Field Type**	**Description**
link_status	CHAR(1)	A code indicating whether the minidisks in the storage group were kept linked or were detached. **D** indicates the minidisks were detached. **L** indicates the minidisks were kept linked. **(blank)** indicates the object is a file space or that the issuer does not have file pool administration authority.
	CHAR(2)	Reserved.

Return Codes and Reason Codes

For lists of the possible return codes from DMSQFPDS, see Appendix D, “Return Codes,” on page 597.

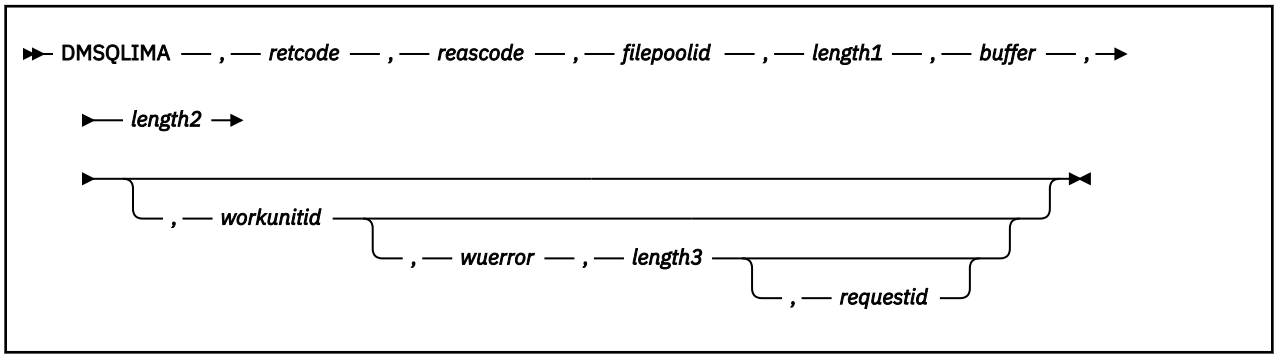
The following table lists the special reason codes returned by DMSQFPDS. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSQFPDS can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	32080	No lock on the specified type of object exists.
WARNING	90270	Output buffer is too small. Some records have been discarded.
ERROR	30000	You do not have the file pool administration authority required to issue a request with the GROUP or ALL parameter.
ERROR	32000	User ID is not enrolled in the file pool.
ERROR	32010	The user ID parameter is missing or is longer than 8 characters.
ERROR	50100	Incorrect storage group number specified. It cannot be less than 2 or greater than the MAXDISKS server parameter.
ERROR	50105	Incorrect storage group number specified. When the FILESPACE or ALL parameter is specified, the storage group number must be 0.
ERROR	50200	Specified storage group does not exist.
ERROR	90210	Extraneous characters in an input parameter.
ERROR	90215	Incorrect buffer length.
ERROR	90300	The object type parameter (GROUP, FILESPACE *filespaceid*, or ALL) or its length is incorrect.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Severity	Reason Code	Description
ERROR	90472	Incorrect request ID specified; it must be 0 or 1.
ERROR	90476	Incorrect file pool ID specified.
ERROR	90540	Incorrect work unit ID was specified.
ERROR	95400	A logical unit of work is being processed for the specified work unit and file pool ID.
ERROR	98700	The file pool does not support DMSQFPDS.

DMSQLIMA - Query Limits



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQLIMA is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQLIMA
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQLIMA routine to return limits information about all the file spaces (SFS users and byte file systems) enrolled in the file pool. This information includes:

- File space ID
- Storage group assigned
- Number of blocks allocated to the file space
- Number of blocks currently used
- Threshold set on blocks used

If you want information for a single file space, use the DMSQLIMU (Query Limits - Single File Space) routine instead.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQLIMA.

reascode
(output, INT, 4) is a variable for the reason code from DMSQLIMA.

filepoolid
(input, CHAR, 1-8) is a variable that identifies the file pool for which you are requesting information. When specifying this parameter, an appended colon is invalid and should not be used.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*filepoolid*).

buffer
(output, CHAR, *length2*) is an area where information is to be returned. It must have a length of at least 32: 8 bytes plus 24 bytes for each enrolled file space. (See the Usage Notes for more information about the *buffer* parameter.)

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

workunitid
(input, INT, 4) is a variable identifying the work unit associated with the DMSQLIMA routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, this indicates that the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. DMSQLIMA and DMSQLIMU (Query Limits - Single File Space) provide the function of the CMS QUERY LIMITS command, which is described in *z/VM: CMS Commands and Utilities Reference*.
2. Rather than have your program deblock the output returned by this routine, you can call the Query Limits - Deblocker (DMSQLIMD) routine to get limits information for one file space from the buffer returned by DMSQLIMA.
3. A variable number of records is returned. The *buffer* parameter must be large enough to hold all the records, one for each enrolled file space. The buffer size can be calculated by multiplying the value returned in number-of-enrolled-file-spaces by 24 and adding 8. For example:

   ```
   number-of-enrolled-file-spaces = 10
   buffersize = (24 * 10) + 8
   buffer size = 248
   ```

   If there are more records available than can fit in the buffer, the buffer is filled with as many records as can fit, and a truncation warning is returned. The number-of-enrolled-file-spaces field (see usage note “8” on page 396) contains the number of records for which information was available. The DMSQLIMA routine can then be reissued with the correct size calculated by using the value returned in the number-of-enrolled-file-spaces field.
4. DMSQLIMA is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSQLIMA causes a noncoordinated commit to be done for the work in the affected file pool.
5. A request ID is returned if the request is to be asynchronous.
6. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.

7. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
8. The output buffer has this format:

FIELD TYPE	DESCRIPTION
Integer(4)	Number of records actually returned
Integer(4)	Number of enrolled file spaces for which there was data
Char(8)	Array record: file space ID
Integer(4)	Array record: storage group number
Integer(4)	Array record: maximum number of committed 4KB blocks allowed for *filespaceid*
Integer(4)	Array record: actual number of 4KB blocks committed for *filespaceid*
Integer(2)	Array record: file space threshold warning percentage for *filespaceid*
Char(2)	Array record: filler
Char(1)	Array record: '0'= SFS file space, '1'= BFS file space
Char(1)	Array record: filler

Return Codes and Reason Codes

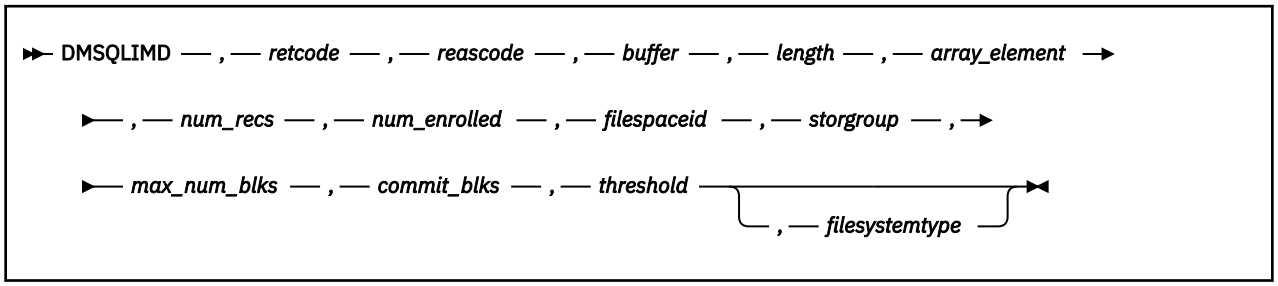
For lists of the possible return codes from DMSQLIMA, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSQLIMA. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSQLIMA can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	32050	No file spaces are enrolled in the file pool.
WARNING	90270	Output buffer is too small. Data has been truncated.
ERROR	30000	Not authorized: file pool administration authority is required.
ERROR	90210	Extraneous characters in input parameter.
ERROR	90215	Invalid buffer length.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid requestid specified, must be 0 or 1.
ERROR	90476	Invalid file pool ID specified.
ERROR	90540	Specified work unit ID is invalid.

DMSQLIMD - Query Limits - Deblocker



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQLIMD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQLIMD
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQLIMD routine to extract limits information about one file space from the buffer returned by the DMSQLIMA (Query Limits) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQLIMD.

reascode
(output, INT, 4) is a variable for the reason code from DMSQLIMD.

buffer
(input, CHAR, *length*) is an area where the Query Limits information was returned from DMSQLIMA.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*). This should have the same value as the length used in DMSQLIMA.

array_element
(input, INT, 4) specifies which element in the array is to be deblocked now.

num_recs
(output, INT, 4) is a variable used to return a number indicating how many elements there are in the array.

num_enrolled
(output, INT, 4) is a variable in which the number of users enrolled is returned.

filespaceid
(output, CHAR, 1-8) is a variable in which is returned the file space ID for which information is being obtained on this call.

storgroup
(output, INT, 4) is a variable in which the storage group number for this file space is returned.

max_num_blks
(output, INT, 4) returns the maximum number of 4KB blocks allowed for this file space.

commit_blks
(output, INT, 4) returns the actual number of 4KB blocks committed by this file space.

threshold
(output, INT, 4) is a variable in which the file space warning threshold percentage is returned.

filesystemtype
(output, CHAR, 1) is a variable in which a value is returned that indicates the file system type of the file space:

0
SFS file space

1
BFS file space

Usage Notes

1. If the value in *num_enrolled* is greater than that in *num_recs*, then the buffer was not large enough to hold all the information available when Query Limits (DMSQLIMA) was called.
2. If the value in *array_element* is greater than that in *num_enrolled*, then the user number requested is considered not valid and an error reason code is returned.
3. If there are no errors, the deblocked information is returned in the appropriate parameters.
4. For a BFS file space, the threshold value is always 100.

Return Codes and Reason Codes

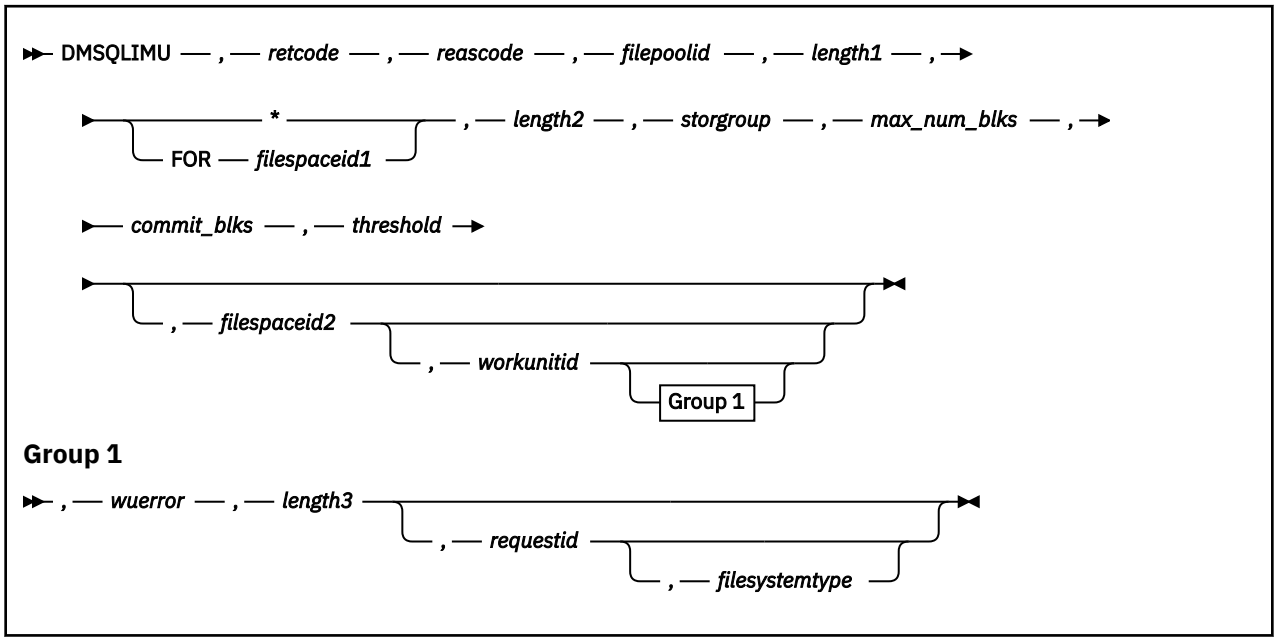
For lists of the possible return codes from DMSQLIMD, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason code returned by DMSQLIMD. ERROR indicates that the request failed. Errors cause return codes 8 or 12.

DMSQLIMD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90221	Array element specified is not valid. It is less than 1 or greater than the returned number of users enrolled.

DMSQLIMU - Query Limits - Single File Space



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQLIMU is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQLIMU
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQLIMU routine to return limits information for one file space (SFS user or byte file system).

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQLIMU.

reascode
(output, INT, 4) is a variable for the reason code from DMSQLIMU.

filepoolid
(input, CHAR, 1-8) is a variable for specifying the name of the file pool (without the colon) containing the information being queried.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*filepoolid*).

***
(input, CHAR, 1) indicates that the file space ID being queried is the APPC/VM ACCESS user ID.

FOR *filespaceid1*
(input, CHAR, 5-12) means that information is to be returned for the one specified file space. Nicknames are not allowed for this parameter.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (* or FOR *filespaceid1*), including delimiting blanks.

storgroup
(output, INT, 4) is a variable in which the storage group number for the specified file space is returned.

max_num_blks
(output, INT, 4) is a variable used to return the maximum number of 4KB blocks allowed for this file space.

commit_blks
(output, INT, 4) is a variable used to return the actual number of 4KB blocks committed by this file space.

threshold
(output, INT, 4) is a variable used to return the threshold percentage for this file space.

filespaceid2
(output, CHAR, 8) is a variable used to return the ID of the file space for which information was returned.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, this indicates that the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request.

filesystemtype
(output, CHAR, 1) is a variable in which a value is returned that indicates the file system type of the file space:

0
SFS file space

1
BFS file space

Usage Notes

1. DMSQLIMU and DMSQLIMA (Query Limits) provide the function of the CMS QUERY LIMITS command, which is described in *z/VM: CMS Commands and Utilities Reference*
2. DMSQLIMU is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSQLIMU causes the work in the affected file pool to be committed without coordination.

3. If the asterisk (*) is specified, the file space ID being queried is the APPC/VM ACCESS user ID. This is the user ID the file pool server uses to identify the connector and to verify authorization.
4. A request ID is returned if the request is to be asynchronous.
5. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
6. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
7. If there are no errors, the limits information for the specified user is returned in the appropriate parameters.

Return Codes and Reason Codes

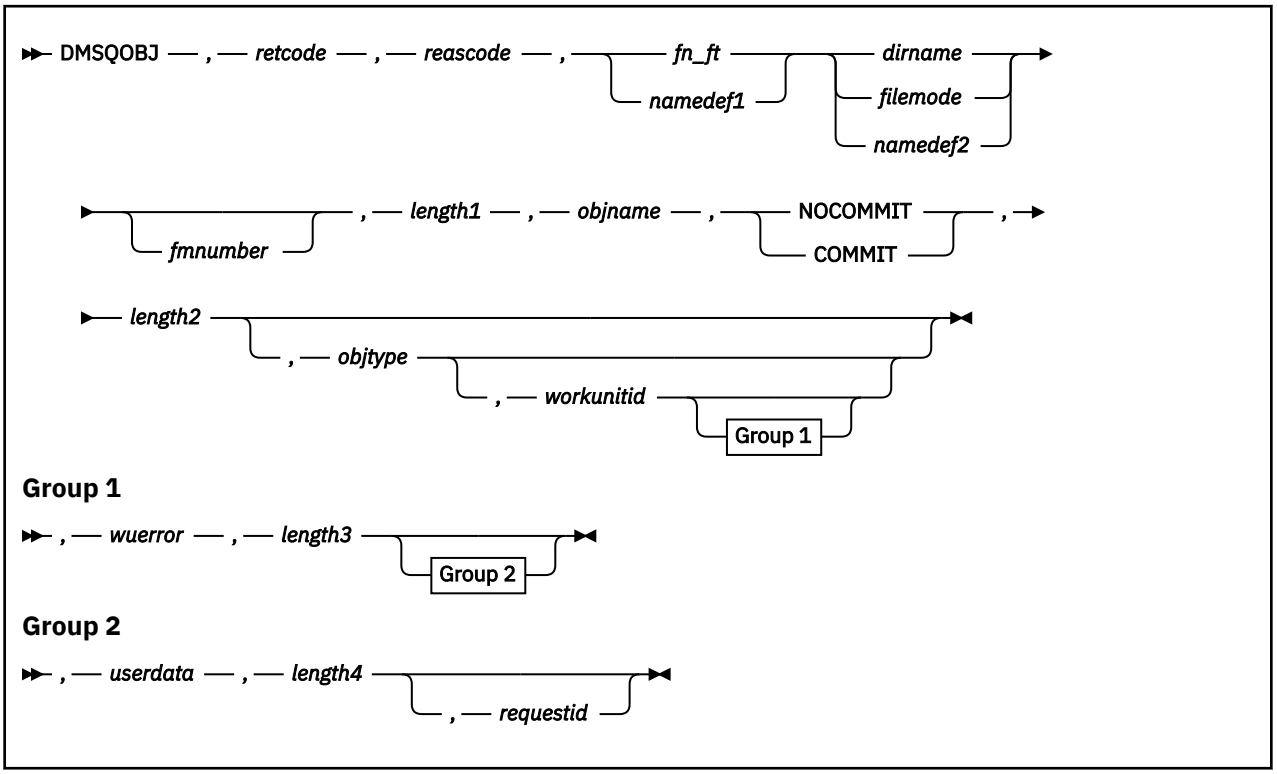
For lists of the possible return codes from DMSQLIMU, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSQLIMU. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSQLIMU can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	32050	The specified file space ID is not enrolled.
ERROR	90210	Extraneous characters in input parameter.
ERROR	90300	Invalid input parameter in CSL parameter list. Specified file space ID is longer than 8 characters.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Specified *requestid* is invalid; must be 0 or 1.
ERROR	90476	Invalid file pool ID specified.
ERROR	90540	Specified work unit ID is invalid.
ERROR	95400	This work unit was already active for the specified file pool when DMSQLIMU was executed.

DMSQOBJ - Query External Object



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSQOBJ is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQOBJ
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSQOBJ" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQOBJ routine to determine the remote name of an external object.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQOBJ.

reascode
(output, INT, 4) is a variable for the reason code from DMSQOBJ.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the external object.

namedef1
(input, CHAR, 1-16) is a variable containing a namedef (a temporary name) for the file name and file type of the external object.

dirname
(input, CHAR, 1-153) is a variable specifying the name of the directory containing the external object.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If this file mode letter is also a one character namedef (*namedef2*), the namedef will be used. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable containing a namedef (a temporary name) for name of the directory containing the external object.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number. It is a character value between '0' and '6', inclusive. If *fmnumber* is specified but does not match the file mode number for the file or external object, a warning is returned. If *fmnumber* is not specified and the file or external object is found, the return code is 0. The *fmnumber* field is ignored for directories.

length1
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*fn_ft* or *namedef1*, *dirname* or *namedef2*, and *fmnumber*).

objname
(output, CHAR, 255) is a variable used to return the remote name.

COMMIT
(input, CHAR, 6) means to keep all changes associated with the work unit. That is, all changes made during a work unit, from either the start of the work unit or from the last COMMIT, are kept.

NOCOMMIT
(input, CHAR, 8) means not to commit changes.

length2
(input, INT, 4) is a variable containing the length of the preceding parameter (COMMIT or NOCOMMIT).

objtype
(output, CHAR, 8) is a variable containing the object type (which identifies the creator of the object).

The format and meaning of the data in this field are defined by the program querying the external object. The first 3 characters, however, are designed to be used as a product identifier. To ensure uniqueness, this identifier is assigned by IBM. Its initial value is zero. For more information about these identifiers, see the information about the ANCHOR macro in the *z/VM: CMS Macros and Functions Reference*.

workunitid
(input, INT, 4) is a variable containing the work unit to be used to go to a file pool. This must be a signed integer variable with a length of 4. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) contains the information necessary to have CMS return extended error information. If it is specified, it must be followed by a length field.

length3
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable containing a string of up to 80 characters of user data to be passed to an external security manager (ESM). The format and meaning of the data is defined by the ESM (see the Usage Notes.)

length4
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, the request is to be asynchronous and CMS generates an integer to identify that request. This integer is placed in *requestid*, which is passed on a later CHECK (DMSCHECK) request.

Usage Notes

1. In order to issue DMSQOBJ, you must have read authority on the directory containing the external object.
2. If your installation does not use an external security manager (ESM), *userdata* is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify passwords for the source and target files.
3. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Query Object operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
4. For an asynchronous request, the return code indicates only whether the request was accepted for processing (rc=0) or was immediately rejected; processing errors can still occur. In contrast, a return code of 0 for a synchronous request means the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSQOBJ, see Appendix D, “Return Codes,” on page 597.

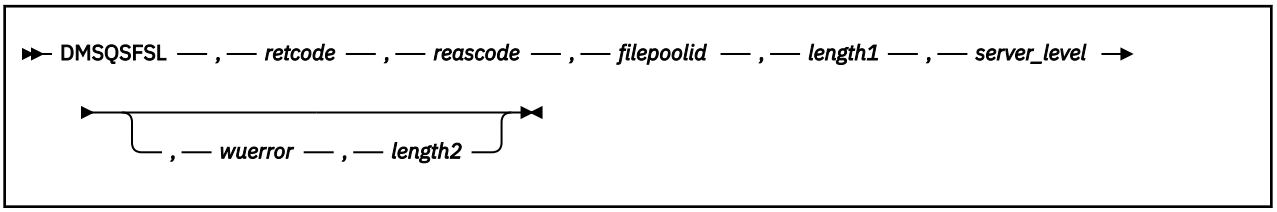
The following table lists the special reason codes returned by DMSQOBJ. ERROR means the request failed, or the request was unsuccessful. Errors cause return code 8 or 12.

DMSQOBJ can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	44000	Not authorized or directory does not exist.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.

Severity	Reason Code	Description
ERROR	90250	File name and file type (or *namedef*) are required, but were not specified.
ERROR	90300	Invalid parameter in CSL parameter list.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least 2 and not more than 4 tokens in the string.
ERROR	90380	Missing parameter in CSL parameter list.
ERROR	90405	Specified object is not an external object.
ERROR	90410	Invalid length specified for one of the character parameters.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fn_ft* parameter is invalid. The file name is longer than eight characters or contains an invalid character.
ERROR	90430	The file type in the *fn_ft* parameter is invalid. The file type is longer than eight characters or contains an invalid character.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fn_ft* parameter.
ERROR	90472	Invalid *requestid*, must be 0 or 1.
ERROR	90500	The specified *dirname* is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90540	Specified *workunitid* is invalid.
ERROR	90590	There is no default file pool currently defined, and file pool ID was not specified as part of the directory name.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.

DMSQSFSL - Query File Pool Server Level



Context

File Pool Administration

Call Format

The format for invoking a CSL routine is language dependent. This CSL routine can be called only called indirectly, through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQSFSL
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQSFSL routine to identify the functional level of the file pool server machine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is the return code from DMSQSFSL.

reascode
(output, INT, 4) is the reason code from DMSQSFSL.

filepoolid
(input, CHAR, 1-8) is a variable containing the name of the file pool, without the colon.

length1
(input, INT, 4) is the length of the *filepoolid* parameter.

server_level
(output,INT,4) is a unique code for the functional level of CMS in which the file pool server is running:

Table 44. CMS Level Codes

Code	Level	Introduced
4	6	VM/SP Release 6
8	7	VM/ESA Version 1 Release 1.0
12	8	VM/ESA Version 1 Release 1.1
16	9	VM/ESA Version 1 Release 2.0
20	10	VM/ESA Version 1 Release 2.1
24	11	VM/ESA Version 1 Release 2.2
28	12	VM/ESA Version 2 Release 1.0

Table 44. CMS Level Codes (continued)		
Code	**Level**	**Introduced**
32	13	VM/ESA Version 2 Release 2.0
36	14	VM/ESA Version 2 Release 3.0
40	15	VM/ESA Version 2 Release 4.0
44	16	z/VM Version 3 Release 1.0
48	17	z/VM Version 4 Release 1.0
52	18	z/VM Version 4 Release 2.0
56	19	z/VM Version 4 Release 3.0
60	20	z/VM Version 4 Release 4.0
64	21	z/VM Version 5 Release 1.0
68	22	z/VM Version 5 Release 2.0
72	23	z/VM Version 5 Release 3.0
76	24	z/VM Version 5 Release 4.0
80	25	z/VM Version 6 Release 1.0
84	26	z/VM Version 6 Release 2.0
88	27	z/VM Version 6 Release 3.0
92	28	z/VM Version 6 Release 4.0
96	29	z/VM Version 7 Release 1.0
100	30	z/VM Version 7 Release 2.0

wuerror
(output, CHAR, *length2*) is an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and will result in an error return code.

Return Codes and Reason Codes

For lists of the possible return codes from DMSQSFSL, see Appendix D, "Return Codes," on page 597.

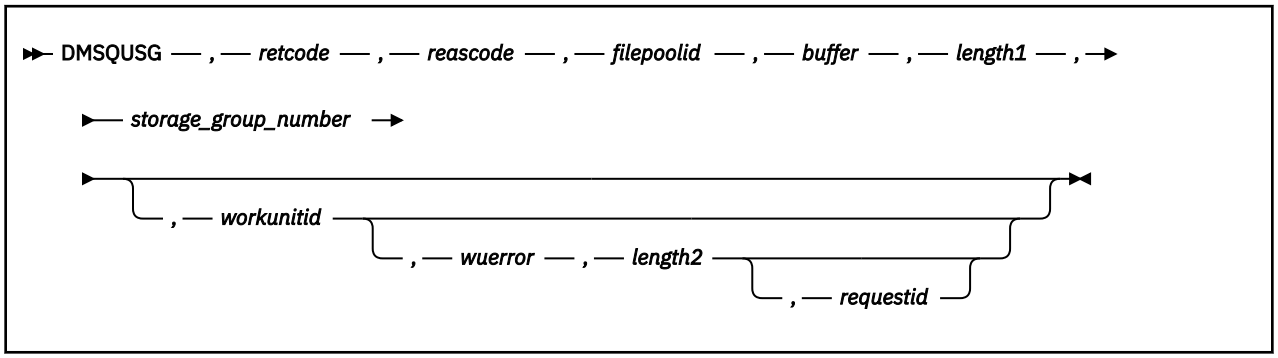
The following table lists the special reason codes returned by DMSQSFSL. ERROR means the request failed. Errors cause return code 8 or 12.

DMSQSFSL can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	57000	The storage group to which this user belongs is being restored.
ERROR	86400	File pool server level information could not be determined, possibly because of corrupted storage.

Severity	Reason Code	Description
ERROR	90210	Extraneous characters in file pool ID.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90476	Incorrect file pool ID specified.

DMSQUSG - Query User Storage Group



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQUSG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQUSG
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSQUSG" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQUSG routine to obtain information about user storage groups in a file pool. You must have file pool administration authority to use this routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQUSG.

reascode
(output, INT, 4) is a variable for the reason code from DMSQUSG.

filepoolid
(input, CHAR, 8) is a variable containing the name of the file pool containing the storage groups you are querying.

buffer
(output, CHAR, *length1*) is a field where information is returned.

length1
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter *buffer*. It must be at least 24: 12 bytes plus an additional 12 bytes for each storage group.

storage_group_number
(input, INT, 2) is a variable containing the storage group number. It is a value between 2 and 32767, and not greater than the MAXDISKS parameter used on the FILESERV GENERATE command for the file pool. If you specify 0, you will receive information for all user storage groups. Specifying 1 is incorrect, because 1 is not a valid user storage group number.

workunitid
(input, INT, 4) is a variable containing the ID of the work unit to be used to go to a file pool. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length2*) contains the information necessary to have CMS return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12, plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request can be asynchronous. CMS returns an integer in *requestid* to identify an asynchronous request. This ID is passed on a later Check (DMSCHECK) request. CMS returns a 1 in *requestid* if the request was completed synchronously and it is not necessary to call DMSCHECK.

Usage Notes

1. Rather than deblock the output returned by this routine, you can call the Query User Storage Group - Deblocker (DMSQUSGD) routine to get one storage group's information from the buffer returned by DMSQUSG.
2. The number of records returned varies. The *buffer* parameter must be large enough to hold all the records, one for each storage group. If more information is available than can fit in the buffer, the buffer is filled and a warning is issued that the information was truncated. The "Number of User Storage Groups" field in the buffer contains the number of records for which information was available. You can calculate the correct size using this value, and reissue DMSQUSG.
3. DMSQUSG is an atomic request. This means that there can be no outstanding work for the file pool on the work unit when this routine is called. When it is finished, DMSQUSG the work done on the work unit for the file pool.
4. The output buffer has this format:

FIELD TYPE	DESCRIPTION
Integer(4)	Number of records actually returned
Integer(4)	Number of user storage groups
Integer(4)	Length of each array record
Integer(2)	Array record: storage group number
Integer(2)	Array record: GROUPTHRESH value
Integer(4)	Array record: total blocks in storage group
Integer(4)	Array record: blocks free in storage group

5. The GROUPTHRESH value is defined by the GROUPTHRESH SFS start-up parameter. It is the same for all user storage groups.
6. The *blocks_free_in_storage_group* value is the number of blocks that are currently unallocated. The *total_blocks_in_storage_group* value minus the *blocks_free* value yields the number of blocks currently allocated. These allocations are for blocks that are committed **plus** blocks that are uncommitted in active work units. The uncommitted allocated blocks may be allocated because of 'shadowing' or because they are new allocations.

7. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
8. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

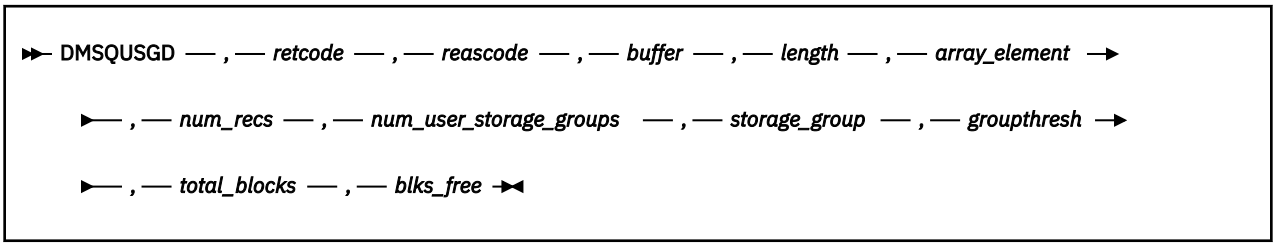
For lists of the possible return codes from DMSQUSG, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSQUSG. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed, or the request was unsuccessful. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSQUSG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	90270	Warning: input buffer is too small.
ERROR	30000	Administrator Authority required.
ERROR	44000	Specified storage group does not exist.
ERROR	50103	Incorrect storage group number specified. Storage group cannot be 1 or less than 0.
ERROR	90215	Incorrect buffer length.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	90476	Incorrect *filepoolid*.
ERROR	90540	Specified *workunitid* incorrect.
ERROR	95400	Logical unit of work is already in process for the specified *workunitid*.

DMSQUSGD - Query User Storage Group - Deblocker



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSQUSGD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQUSGD
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQUSGD routine to obtain specific information about user storage groups in a file pool from the array of information returned by the DMSQUSG (Query User Storage Group) routine. You must have file pool administration authority to use this routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQUSGD.

reascode
(output, INT, 4) is a variable for the reason code from DMSQUSGD.

buffer
(input, CHAR, *length*) is the buffer in which the information was returned from DMSQUSG. It must be the same length as the buffer returned from DMSQUSG.

length
(input, INT, 4) is a signed integer variable containing the length of the preceding character parameter *buffer*. It must have the same value as the length used in DMSQUSG.

array_element
(input, INT, 4) specifies which element in the array from the buffer is to be deblocked. Specify 1 for the first element.

num_recs
(output, INT, 4) returns the number of elements in the array.

num_user_storage_groups
(output, INT, 4) returns the number of user storage groups about which information was requested. If *storage_group* on the original DMSQUSG request was a storage group number, then *num_user_storage_groups* is 1. If *storage_group* was 0, then the number of user storage groups in the file pool is returned.

storage_group
(output, INT, 2) is a variable containing the number of the storage group.

groupthresh
(output, INT, 2) is a variable containing the GROUPTHRESH value. (This is a SFS start-up parameter and is the same for all storage groups.)

total_blocks
(output, INT, 4) is a variable in which the total number of 4K blocks in the storage group is returned.

blks_free
(output, INT, 4) is a variable in which the number of 4K blocks free in the storage group is returned.

Usage Notes

1. If the value in *num_user_storage_groups* is greater than *num_recs*, then the buffer was not large enough to hold all the information returned by DMSQUSG.
2. If the value in *array_element* is greater than *num_recs*, then the array element number is considered invalid and an error reason code is returned.
3. If no errors occur, the deblocked information is returned in the appropriate parameters.

Return Codes and Reason Codes

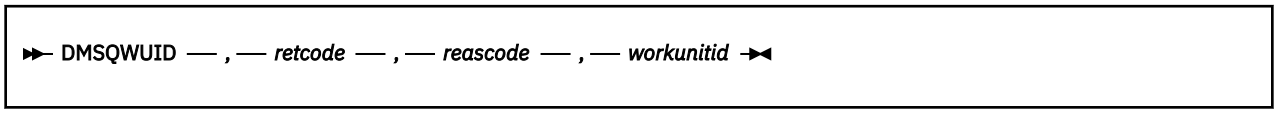
For lists of the possible return codes from DMSQUSGD, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason code returned by DMSQUSGD. ERROR means the request failed, or the request was unsuccessful. Errors cause return code 8 or 12.

DMSQUSGD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90217	The requested element is less than or equal to zero, or the requested element is greater than the number of records being returned in the buffer.

DMSQWUID - Query Work Unit ID



Context

Work Unit Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSQWUID is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSQWUID
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSQWUID routine to find out what the current default work unit identifier is.

Parameters

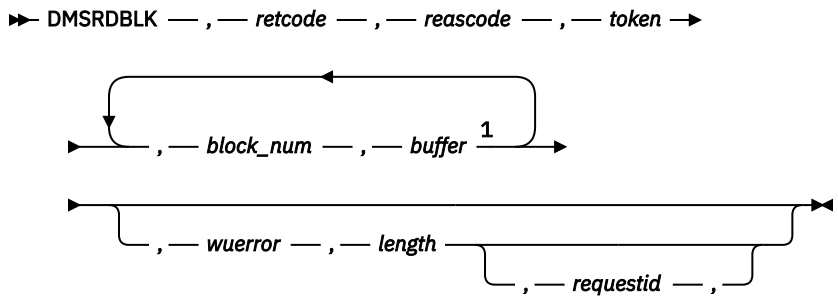
Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSQWUID. There are no error conditions returned by this routine, so the return code should always be 0.

reascode
(output, INT, 4) is a variable for the reason code from DMSQWUID. There are no error conditions returned by this routine, so the reason code should always be 0.

workunitid
(output, INT, 4) is a variable for returning the current default work unit ID.

DMSRDBLK - Read Blocks



Notes:

[1] Specify one to seven pairs of block numbers and buffers.

Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSRDBLK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRDBLK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRDBLK routine to read blocks from a file that has been opened with the DMSOPBLK (Open Blocks) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRDBLK.

reascode
(output, INT, 4) is a variable for the reason code from DMSRDBLK.

token
(input, CHAR, 8) contains the *token* returned by Open Blocks (DMSOPBLK) when the file was opened.

block_num
(input, INT, 4) is the block number to be read.

Up to seven block numbers can be specified. If any subsequent parameters are used, specify the entire list of seven block numbers and buffers, and set a block number to 0 to indicate the end of the list of block numbers to be read. All following block numbers are ignored.

buffer
(output, CHAR, 4K) is an area into which the 4KB block is read. Up to 7 buffers can be specified. If the first block number specified is 5, the first buffer contains the 4 K bytes of data of block 5.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If on return the request ID is still 1, no server call was needed. It is be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. In the list of block number—buffer pairs, any nonzero block number must have a corresponding buffer with a length of 4 K bytes.
2. The number of block numbers must equal the number of buffers. The block numbers and buffer areas are treated as pairs.
3. A request ID is returned if the request is to be asynchronous.
4. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
6. Using a block number more than once in the same request produces an error reason code.

Return Codes and Reason Codes

For lists of the possible return codes from DMSRDBLK, see Appendix D, “Return Codes,” on page 597.

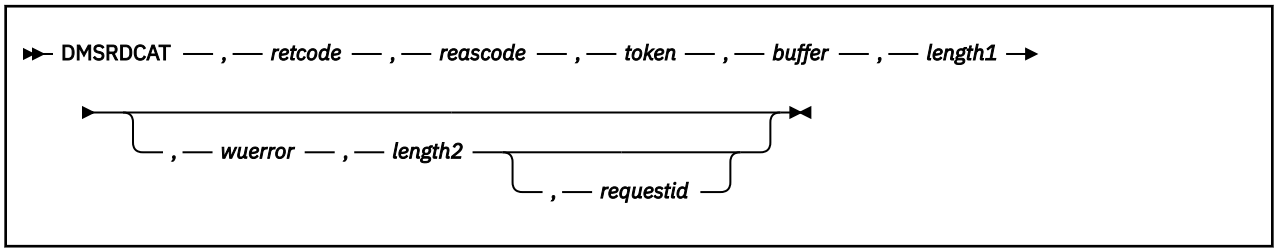
The following table lists the special reason codes returned by DMSRDBLK. ERROR indicates that the request failed. Errors cause return codes 8 or 12.

DMSRDBLK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	10000	• System error. File is not open. • You tried to read a file that was opened with an intent of CREATEMIG.
ERROR	54000	Attempt to read logical block number not associated with the file.
ERROR	71200	Either SFS made an error in accessing the file, or you specified the same block number more than once.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Request ID must be 0 or 1.
ERROR	90488	Invalid number of buffers specified.

Severity	Reason Code	Description
ERROR	90490	Invalid number of blocks specified.
ERROR	95700	No open file pool object found for the specified token.

DMSRDCAT - Read Catalog



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSRDCAT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRDCAT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRDCAT routine to return file pool catalog information to the caller. The information returned depends on the object for which the catalog was opened.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRDCAT.

reascode
(output, INT, 4) is a variable for the reason code from DMSRDCAT.

token
(input, CHAR, 8) contains the *token* returned from the Open Catalog (DMSOPCAT) for this file. Note that the token is returned on Open Catalog as part of a buffer, not as a separate variable.

buffer
(output, CHAR, *length1*) is a variable containing the buffer into which the catalog information is to be returned. On return, the first full word of the buffer will contain the length of the data. This is followed by the data itself. This format must be used on any future Write Catalog request.

length1
(input, INT, 4) is a variable containing the length of the preceding character parameter (*buffer*).

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the wuerror parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, *requestid* is still 1, no server call was needed. It is not necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. Only the amount of catalog data that fits into the buffer is returned. Additional Read Catalog (DMSRDCAT) requests may have to be done to read the rest of the catalog data. An end-of-data reason code is returned when the buffer either contains the final records or is empty (all records have already been returned). Your program should be coded to handle both cases.
2. It is recommended that the buffer length be a multiple of 4KB. The minimum length specified must be at least 296 bytes. A single DMSRDCAT request will return at most 120KB of catalog data in the buffer. Several calls to DMSRDCAT are necessary to fill a buffer larger than 120KB.

   When there are several DMSRDCAT requests, the last catalog record in the buffer is a complete record. A record is never split between two requests. Thus, you may see some unused space at the end of the buffer.
3. Your program needs to be sensitive to the case where a set of records should be returned for an object, but only a subset of the appropriate records is returned due to inconsistent catalogs. An example of this is where an OBJECTCAT for an alias is returned without its FQFN record. In this case your code would have to know to pass over the OBJECTCAT for the alias. Note that this condition would happen only with corrupted catalog data, but must be considered when writing an application.
4. A request ID is returned if the request is to be asynchronous.
5. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
6. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
7. The catalog data is placed in the buffer, along with the length of the data.

   The following usage notes apply only to SFS file spaces:

   a. If the object is a storage group, the catalog information and authorizations for all file spaces, directories, aliases and files associated with the specified storage group are returned.

      For an opened file space, the catalog information and authorizations for all directories, aliases, and files associated with the specified file space are returned.

      For an opened directory, the catalog information and authorizations for all files within the specified directory are returned.

   b. When a user is enrolled in a file pool, the file pool server creates a SPACECAT record for that user. The SPACECAT record indicates how much space, if any, has been assigned to that user, what storage group the user has been assigned to, and the percentage at which the user should be warned that the assigned space is almost consumed.

      If the open intent was for READEXT on a storage group or file space, following the SPACECAT record will be the "external authorization" records. Each object in the file pool that this user has explicit authority to will be represented by an AUTHCAT record (CATTYPE=G) followed by an FQFN record containing the fully qualified name of the object.

      Note that explicit authority does not include file pool administration authority or public authority or authority to files within the user's own file space.

      If the open intent was for READEXT on a directory, no "external authorization" records will be returned.

The user also is assigned a top directory, represented by a DIRCAT record and an OBJECTCAT record. A unique ID is associated with this directory and is saved in the directory object ID field (DOID) of DIRCAT and the object ID field (OID) in OBJECTCAT. This ID is also stored in the SPACECAT record in the user object ID field (USEROID). The server represents all directories by a DIRCAT record and OBJECTCAT record pair.

To support a hierarchy of directories, every object associated with a user has a parent, except the top directory. The parent directory object ID (PDOID) is stored in the OBJECTCAT record.

All aliases, files, and directories have an OBJECTCAT record. An alias can be associated with its base file through the base file object ID field (BFOID). The file attributes are contained in the OBJECTCAT record of a file and are replicated in the OBJECTCAT record of an alias.

The OBJECTCAT record for a file also contains the first 8 data block numbers in the DATABLOCKLIST field. The numbers are relative to the beginning of the file pool.

If a file has more than eight 4KB blocks of data, these additional blocks are represented by DBLCAT records. The DBLCAT records for a file are logically related to the file's OBJECTCAT record by the object ID field (OID) in both records.

When any authorizations are granted on either a file or directory, an AUTHCAT record is created to represent the authorization. An AUTHCAT record contains the authority granted, the object it is granted on, the user it is granted to, and the owner of the object.

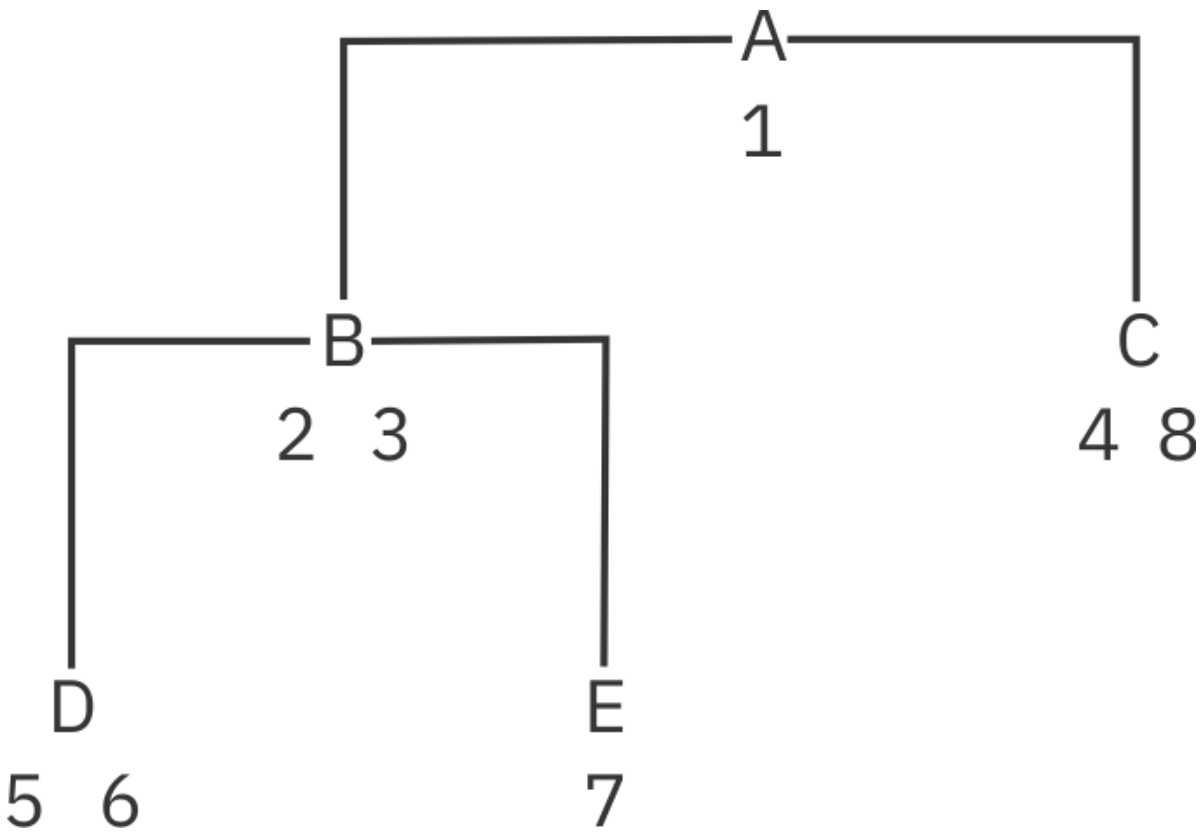
If the open intent was READEXT, following the AUTHCAT records for a base file, each alias in the file pool that points to this base file and is not part of this file space will be represented by an FQFN record with CATTYPE=H. This 'H' type FQFN record contains the fully qualified file name of the external alias.

Following an OBJECTCAT record whose STATUS field is 'V' (alias) will be the FQFN record, which contains the fully qualified file name of the base file.

An ACAT record will follow the OBJECTCAT record whose STATUS field is 'J' (unresolved alias). The ACAT record contains the fully qualified file name of the base file.

An EOCAT record is returned for external objects.

c. When returning information about an SFS file space or storage group, the DMSRDCAT routine first returns all information related to a directory, and then about each of its subdirectories. For example, suppose you are reading information about this file space:



For this user, A, B, C, D, E are directories.

The numbers are files that are in the directory that they immediately follow. All files are base files except for file 3, which is an alias, file 4, which is an unresolved alias, and file 8, which is an external object.

If you open the file space for READ or READEXT, DMSRDCAT returns catalog records in the following sequence. Records returned only for READEXT are indicated by an asterisk (*).

Table 45. Catalog Record Sequence Returned for an SFS File Space Opened for READ or READEXT

CATTYPE	**Contents**
S	SPACECAT for user
G *	AUTHCAT for directory ZACK.ZZ
F *	FQFN of ZACK's directory ZZ
D	DIRCAT for directory A
O	OBJECTCAT for directory A
A	AUTHCAT for directory A
O	OBJECTCAT for file 1
A	AUTHCAT for file 1
A	AUTHCAT for file 1
O	OBJECTCAT for directory A.B
A	AUTHCAT for directory A.B
O	OBJECTCAT for directory A.C
A	AUTHCAT for directory A.C

Table 45. Catalog Record Sequence Returned for an SFS File Space Opened for READ or READEXT (continued)

CATTYPE	Contents
D	DIRCAT for directory A.B
O	OBJECTCAT for file 2
A	AUTHCAT for file 2
O	OBJECTCAT for alias 3
F	FQFN of base file for 3
O	OBJECTCAT for directory A.B.D
A	AUTHCAT for directory A.B.D
O	OBJECTCAT for directory A.B.E
A	AUTHCAT for directory A.B.E
D	DIRCAT for directory A.B.D
O	OBJECTCAT for file 5
A	AUTHCAT for file 5
O	OBJECTCAT for file 6
X	DBLCAT for file 6
D	DIRCAT for directory A.B.E
O	OBJECTCAT for file 7
X	DBLCAT for file 7
A	AUTHCAT for file 7
D	DIRCAT for directory A.C
O	OBJECTCAT for unresolved alias 4
R	ACAT for unresolved alias 4
O	OBJECTCAT for file 8
A *	AUTHCAT for file 8
H *	FQFN of ZACK's file 99
E	EOCAT for file 8

d. When the DMSOPCAT intent is FILEATTR, this information is returned for different objects:

Object
Information

Directory
A DIRCAT record (CATTYPE=D) followed by an OBJECTCAT (CATTYPE=O) record for *each file, alias, external object, and subdirectory* in the directory

File space
A SPACECAT record (CATTYPE=S) followed by a series of DIRCAT/OBJECTCAT groups for *each directory* in the file space.

Storage groups
A series (in no particular order) of SPACECAT/DIRCAT/OBJECTCAT groups for *each file space* in the storage group (or all storage groups).

The following usage notes apply only to BFS file spaces:

a. A BFS file space will be represented by one SPACECAT record. The SPACECAT record indicates how much space has been assigned to that BFS file space and what storage group the BFS file space has been assigned to. The SPACECAT record will indicate that the file space is a BFS file space. It contains a file space ID, or USEROID.

   The top or root directory in a BFS file space will be represented by a DIRCAT record. This is the only directory in the hierarchy that is reflected in DIRCAT. The SPACECAT record is related to its top directory by the USEROID.

   Object names (path components) will be recorded in two catalogs. The first is NAMECAT. There is exactly one NAMECAT record for each path component name defined. Each NAMECAT record will contain the first 32 bytes of the name. If the name is longer than 32 bytes, the rest of it will be contained in the NAMECAT overflow catalog (NOVCAT). Notice that name delimiters (/ and ending null character) are not stored in NAMECAT or NOVCAT.

   NAMECAT records are related to SPACECAT by USEROID. Each object is assigned an object ID (OID), which relates the object to NAMECAT, OBJECTCAT, and any DBLCAT or EOCAT records. NAMECAT records are also assigned name IDs (NID). NID is a file space unique number that the server assigns to the name. NID relates the NAMECAT record to the NOVCAT records (if any).

   To support a hierarchy of directories, every object associated with a user has a parent directory, except the top directory. The parent directory object ID (PDOID) is stored in the NAMECAT record.

   Note that, unlike SFS, the hierarchy of directories is not reflected in OBJECTCAT. The PDOID of each OBJECTCAT contains the USEROID of the top directory.

   The objects in a BFS file space are directories, regular files, and a variety of special files. Each object is defined by exactly one OBJECTCAT record. In addition, regular files may be defined by one or more DBLCAT records. Symbolic and external links are additionally defined by one or more EOCAT records.

   All BFS objects have at least one "hard" link, or name, associated with them. Hard links are defined by exactly one NAMECAT record and possibly one or more NOVCAT records.

   Hard Links do not have an OBJECTCAT record of their own, but are related to the OBJECTCAT record of the file or directory they are "linked to". Any number of hard links can be related to the same OBJECTCAT record.

   Large files in a BFS file space will be represented by DBLCAT records, just as it happens in SFS file spaces.

   EOCAT records will be used to store the contents of a symbolic or external link. Because EOCAT records are fixed-length, several records may be required to represent one symbolic or external link.

   There are no AUTHCAT records associated with BFS file spaces. The authorization information for BFS objects is contained in the OBJECTCAT record in the ST_MODE field.

   Aliases do not exist in BFS file spaces.

b. When returning information about a BFS file space, the DMSRDCAT routine returns the SPACECAT record for the file space, followed by the DIRCAT, OBJECTCAT, and NAMECAT records for the top directory, followed by all of the NAMECAT records for the file space. The order of the NAMECAT records is by level (see diagram).

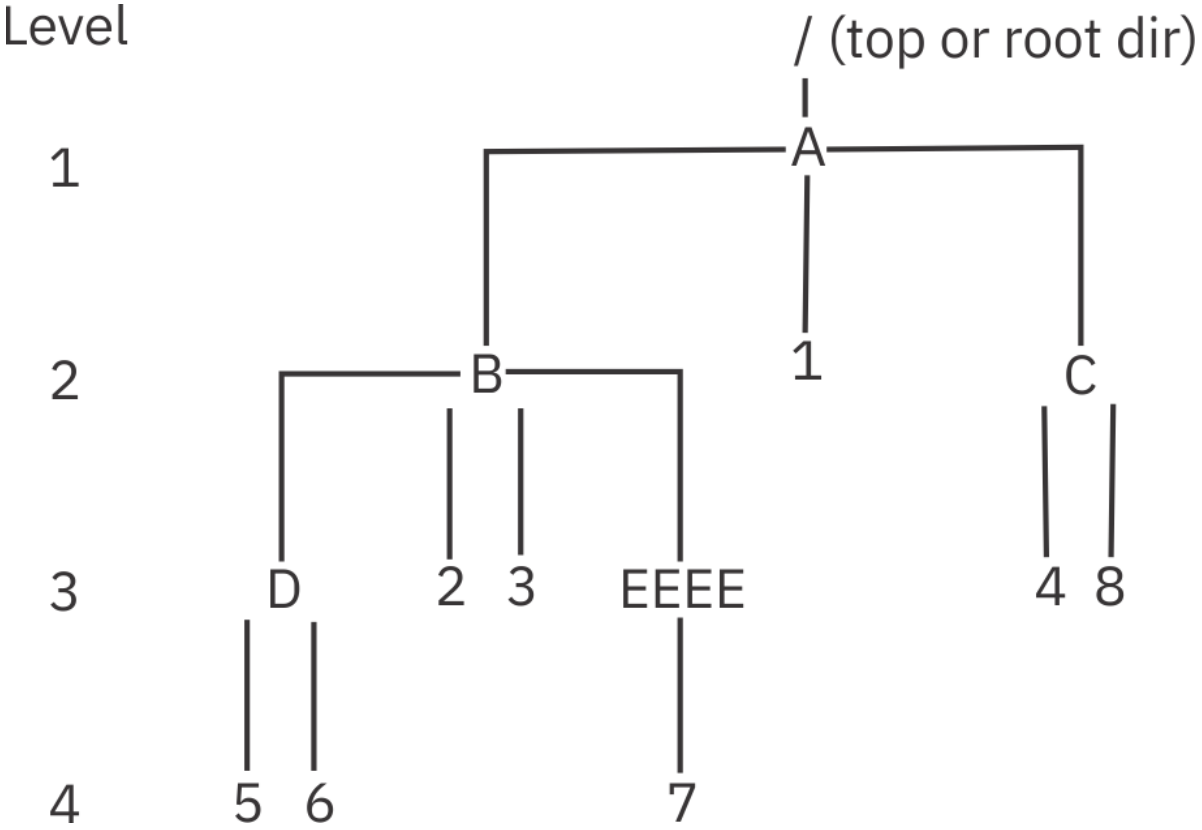
   The OBJECTCAT records for all objects follow. The order of the OBJECTCAT records is alphabetically, by FN FT, which is the CMS short name. Note that the hierarchy of directories is not reflected in OBJECTCAT. The PDOID field of each OBJECTCAT record contains the USEROID of the top directory.

   The CMS short name is assigned as follows: FN is the character representation of the INO, the file space unique number assigned to the object, and FT is always '0 '.

   DBLCAT and EOCAT records follow the OBJECTCAT record as needed.

When returning information about a storage group, each file space in the storage group is returned as above, in no particular order.

When returning information about a directory, all catalog information pertaining to the specified directory is returned. For example, suppose you are reading information about this file space:



For this user, A, B, C, D, EEEE are directories.

The numbers are files that are in the directory that they immediately follow. 8 is a symbolic link. 3 and 1 are two hard links to file 1.

For illustrative purposes, assume the length of a NAMESEG is 3 rather than 32.

If you open the file space for read, DMSRDCAT returns catalog records in the following sequence.

Table 46. Catalog Record Sequence Returned for a BFS File Space Opened for READ

CATTYPE	Contents
S	SPACECAT for BFS file space
D	DIRCAT for the top directory (/)
O	OBJECTCAT for the top directory (/)
N	NAMECAT for the top directory (/)
N	NAMECAT for directory /A
N	NAMECAT for directory /A/B
N	NAMECAT for directory /A/C
N	NAMECAT for file /A/1
N	NAMECAT for directory /A/B/D
N	NAMECAT for directory /A/B/EEEE

Table 46. Catalog Record Sequence Returned for a BFS File Space Opened for READ (continued)	
CATTYPE	**Contents**
V	NOVCAT for directory /EEEE
N	NAMECAT for file /A/B/2
N	NAMECAT for file /A/B/3
N	NAMECAT for file /A/C/4
N	NAMECAT for file /A/C/8
N	NAMECAT for file /A/B/D/5
N	NAMECAT for file /A/B/D/6
N	NAMECAT for file /A/B/EEEE/7
The remaining OBJECTCAT, DBLCAT, and EOCAT records are returned in the numerical order of their CMS short names within the BFS file space. Also note that only one OBJECTCAT record is returned for the object represented by links /A/1 and /A/B/3.	

c. When the DMSOPCAT intent is FILEATTR, this information is returned for different objects:

Object
Information

Directory
A DIRCAT record (CATTYPE=D) followed by an OBJECTCAT (CATTYPE=O) record for each object in the directory.

File space
A SPACECAT record (CATTYPE=S) followed by an OBJECTCAT record for each object in the file space.

Storage groups
A series (in no particular order) of SPACECAT/OBJECTCAT groups for *each file space* in the storage group (or all storage groups).

8. The formats of the catalog records are shown in Tables Table 47 on page 425 through Table 57 on page 434. Fields that apply only to BFS file spaces are indicated by an asterisk (*).

Table 47. SPACECAT Record (CATTYPE = S), Record Length = 61 bytes		
Field Name	**Field Type**	**Description**
CATTYPE	CHAR(1)	Catalog Type
USERID	CHAR(8)	User identifier of the directory owner
STORAGE	INTEGER(4)	Maximum committed 4KB file blocks
STORAGEUSED	INTEGER(4)	Committed 4KB file blocks
STHRESH	INTEGER(2)	Warning threshold percentage
USEROID	CHAR(8)	Internal user identifier
STORAGEGROUP	INTEGER(2)	Storage group number
HIGHINO *	INTEGER(4)	High OBJECTCAT INO value
HIGHNID *	INTEGER(4)	High NAMECAT NID value
FLAGS	CHAR(1)	File space attributes indicated by individual bits: '1xxx xxxx'=BFS file space; '0xxx xxxx'=SFS file space.

Table 47. SPACECAT Record (CATTYPE = S), Record Length = 61 bytes (continued)		
Field Name	**Field Type**	**Description**
RESERVED01	CHAR(7)	Reserved
RESERVED02	CHAR(8)	Reserved
RESERVED03	CHAR(8)	Reserved

Table 48. DIRCAT Record (CATTYPE = D), Record Length = 163 bytes		
Field Name	**Field Type**	**Description**
CATTYPE	CHAR(1)	Catalog Type
DIROWNER	CHAR(8)	Directory Owner
N1	CHAR(16)	Directory Name1
N2	CHAR(16)	Directory Name2
N3	CHAR(16)	Directory Name3
N4	CHAR(16)	Directory Name4
N5	CHAR(16)	Directory Name5
N6	CHAR(16)	Directory Name6
N7	CHAR(16)	Directory Name7
N8	CHAR(16)	Directory Name8
DOID	CHAR(8)	Directory Object ID
USEROID	CHAR(8)	Internal User ID
STORAGEGROUP	INTEGER(2)	Storage Group of DIROWNER
DIRATTS	CHAR(1)	Directory attributes indicated by individual bits: '1xxx xxxx'=DIRCONTROL; '0xxx xxxx'=FILECONTROL; 'x1xx xxxx'=data space eligible; 'x0xx xxxx'=not data space eligible; 'xx0x' xxxx'=SFS file space; 'xx1x xxxx'=BFS file space.
GRANTS	CHAR(1)	Granted authority indicator: '0'=no authorizations granted; '1'=Public Read; '2'=Public Read plus individual grants; '3'=Public Write; '4'=Public Write plus individual grants; '5'=individual grants only (no public).
RESERVED01	CHAR(6)	Reserved

Table 49. OBJECTCAT Record for SFS (CATTYPE = O), Record Length = 216 bytes		
Field Name	**Field Type**	**Description**
CATTYPE	CHAR(1)	Catalog Type
DIROWNER	CHAR(8)	Directory Owner
PDOID	CHAR(8)	Parent Directory OID

Table 49. OBJECTCAT Record for SFS (CATTYPE = O), Record Length = 216 bytes (continued)

Field Name	Field Type	Description
FN	CHAR(8)	CMS file name
FT	CHAR(8)	CMS file type
OWNER	CHAR(8)	Owner of the base file identified by BFOID
TYPE	CHAR(1)	'D'=directory; 'X'=file, alias, or external object; 'U'=unallocated.
FMN	CHAR(1)	CMS File Mode Number
SCID	INTEGER(4)	Status Change Identifier
RECFM	CHAR(1)	CMS file record format: 'F'=fixed; 'V'=variable; 'D'=directory; (blank)=external object or unresolved alias (status J).
OVERFLOW	CHAR(1)	DBLCAT overflow indicator
LRECL	INTEGER(4)	CMS file logical record length
LEVEL	INTEGER(2)	Committed level of the file
STORAGEGROUP	INTEGER(2)	Storage Group of Object
FILEPTR	INTEGER(4)	1st (top) logical 4KB file block
DATABLOCKUSED	INTEGER(4)	Current number of file blocks
RECORDS	INTEGER(4)	Number of file records
PTRLEVEL	INTEGER(1)	Level of Pointer Blocks
FILEFLAGS	CHAR(1)	Directory attributes indicated by individual bits: '1xxx xxxx'=BFS file space; '0xxx xxxx'=SFS file space; 'x1xx xxxx'=20yy century bit for DATE (date last modified); 'x0xx xxxx'=19yy century bit for DATE (date last modified); 'xxx1 xxxx'=file is migrated; 'xxxx 1xxx'=DIRCONTROL; 'xxxx 0xxx'=FILECONTROL; 'xxxx x1xx'=file is nonrecoverable; 'xxxx x0xx'=file is recoverable; 'xxxx xx1x'=file is update-in-place; 'xxxx xx0x'=file is not update-in-place; 'xxxx xxx1'=20yy century bit for DATEREF (date of last reference); 'xxxx xxx0'=19yy century bit for DATEREF (date of last reference);
DATE	CHAR(3)	Local date (yymmdd) last modified
TIME	CHAR(3)	Local time (hhmmss) last modified

Table 49. OBJECTCAT Record for SFS (CATTYPE = O), Record Length = 216 bytes (continued)

Field Name	Field Type	Description
OID	CHAR(8)	Object Identifier
BFOID	CHAR(8)	Base File Identifier
STATUS	CHAR(1)	Status of entry: 'A'=file; 'V'=alias; 'E'=erased alias; 'R'=revoked alias; 'D'=directory; 'I'=unresolved alias (base not restored); 'J'=unresolved alias (base not restored); 'K'=external object; 'U'=unallocated row.
GRANTS	CHAR(1)	Granted authority indicator: '0'=no authority has been granted; '1'=Public Read; '2'=Public Read plus individual grants; '3'=Public Write; '4'=Public Write plus individual grants; '5'=individual grants only (no public).
MAXBLOCK	INTEGER(4)	Maximum file block number
DATABLOCKLIST	CHAR(32)	8-file data block identifiers
USEROID	CHAR(8)	Internal User ID
BUSEROID	CHAR(8)	Internal User ID of Base
BPDOID	CHAR(8)	Parent Directory ID of the Base File
BSTORAGEGROUP	INTEGER(2)	Storage Group of Base File
BSCID	INTEGER(4)	Status Change Identifier (SCID) of Base File
	CHAR(8)	reserved
CHGDATE_ CENTURY	CHAR(1)	Century byte for LAST_CHANGE_DATE (date of last change) X'19' = 19yy X'20' = 20yy
LAST_CHANGE_DATE	CHAR(3)	UTC date of last change
	CHAR(1)	reserved
LAST_CHANGE_TIME	CHAR(3)	UTC time of last change
DRA1	CHAR(8)	DRA1 (DFSMS/VM-related attribute)
DRA2	CHAR(8)	DRA2 (DFSMS/VM-related attribute)
DRA3	CHAR(8)	DRA3 (DFSMS/VM-related attribute)
	INTEGER(4)	reserved

Table 49. OBJECTCAT Record for SFS (CATTYPE = O), Record Length = 216 bytes (continued)		
Field Name	**Field Type**	**Description**
CREATIONDATE_ CENTURY	CHAR(1)	Century byte for CREATIONDATE (date of creation). Either X'19' or X'20'.
CREATIONDATE	CHAR(3)	Date of file creation in Coordinated Universal Time (UTC).
	CHAR(1)	reserved
CREATIONTIME	CHAR(3)	Time of file creation in Coordinated Universal Time (UTC).
DATEREF	CHAR(3)	Date of last reference in Coordinated Universal Time (UTC).

Table 50. OBJECTCAT Record for BFS (CATTYPE = O) Record Length = 216 bytes		
Field Name	**Field Type**	**Description**
CATTYPE	CHAR(1)	Catalog Type
FSNAME	CHAR(8)	File Space Name
PDOID	CHAR(8)	User OID of top directory
FN	CHAR(8)	CMS Short Name
FT	CHAR(8)	'0 '
OWNER	CHAR(8)	File Space Name
TYPE	CHAR(1)	'D'=directory; 'X'=file or link; 'U'=unallocated.
RESERVED01	CHAR(1)	Reserved
SCID	INTEGER(4)	Status Change Identifier
RESERVED02	CHAR(1)	Reserved
OVERFLOW	CHAR(1)	DBLCAT overflow indicator
ST_INO	INTEGER(4)	File Serial Number
LEVEL	INTEGER(2)	Committed level of the file
STORAGEGROUP	INTEGER(2)	Storage Group of Object

Table 50. OBJECTCAT Record for BFS (CATTYPE = O) Record Length = 216 bytes (continued)

Field Name	Field Type	Description
ST_NLINK	INTEGER(4)	Number of Links
DATABLOCKUSED	INTEGER(4)	Current number of file blocks
ST_UID	INTEGER(4)	User ID of file owner
RESERVED03	CHAR(1)	Reserved
FILEFLAGS	CHAR(1)	Directory attributes indicated by individual bits: 'xxx1 xxxx'=file is migrated; '1xxx xxxx'=BFS file space.
RESERVED04	CHAR(3)	Reserved
RESERVED05	CHAR(3)	Reserved
OID	CHAR(8)	Object Identifier
RESERVED06	CHAR(8)	Reserved
STATUS	CHAR(1)	Status of entry: 'A'=BFS regular file; 'B'=block special file; 'C'=character special file; 'D'=directory; 'X'=external link; 'P'=FIFO; 'L'=symbolic link; 'U'=unallocated row.
RESERVED07	CHAR(1)	Reserved
ST_MODE	INTEGER(4)	File mode and permission bits. This field is mapped by the BPXYMODE macro. See the *z/VM: OpenExtensions Callable Services Reference*.
DATABLOCKLIST	CHAR(32)	8-file data block identifiers
USROID	CHAR(8)	User OID
RESERVED08	CHAR(8)	Reserved
ST_SIZE	INTEGER(8)	File size in bytes
RESERVED09	CHAR(2)	Reserved
ST_ATIME	INTEGER(4)	Time of last access
ST_MTIME	INTEGER(4)	Time of last data modification
ST_CTIME	INTEGER(4)	Time of last status change
AUDIT_FLAGS	CHAR(8)	Audit flags

Table 50. OBJECTCAT Record for BFS (CATTYPE = O) Record Length = 216 bytes (continued)		
Field Name	**Field Type**	**Description**
DRA1	CHAR(8)	Reserved
DRA2	CHAR(8)	Reserved
DRA3	CHAR(8)	Reserved
ST_GID	INTEGER(4)	Group ID of the group of the file
CREATIONTIME	INTEGER(4)	Time of file creation.
MAJORDEV	CHAR(2)	Major Device Num
MINORDEV	CHAR(2)	Minor Device Num
RESERVED10	CHAR(3)	Reserved

Table 51. DBLCAT Record (CATTYPE = X) Record Length = 93 bytes		
Field Name	**Field Type**	**Description.**
CATTYPE	CHAR(1)	Catalog Type
OID	CHAR(8)	Internal Object (File) Identifier
SEQ	INTEGER(4)	Sequence Number
DATABLOCKLIST	CHAR(80)	20-file data block identifiers

Table 52. FQFN Record (CATTYPE = F or H), Record Length = 153 bytes		
Field Name	**Field Type**	**Description.**
CATTYPE	CHAR(1)	Catalog Type
DIROWNER	CHAR(8)	Directory Owner
N1	CHAR(16)	Directory Name1
N2	CHAR(16)	Directory Name2
N3	CHAR(16)	Directory Name3
N4	CHAR(16)	Directory Name4

Table 52. FQFN Record (CATTYPE = F or H), Record Length = 153 bytes (continued)		
Field Name	**Field Type**	**Description.**
N5	CHAR(16)	Directory Name5
N6	CHAR(16)	Directory Name6
N7	CHAR(16)	Directory Name7
N8	CHAR(16)	Directory Name8
FN	CHAR(8)	CMS file name
FT	CHAR(8)	CMS file type

Table 53. AUTHCAT Record (CATTYPE = A or G), Record Length = 26 bytes		
Field Name	**Field Type**	**Description.**
CATTYPE	CHAR(1)	Catalog Type
DIROWNER	CHAR(8)	Directory Owner (Grantor)
OID	CHAR(8)	Internal Directory or File ID
GRANTEE	CHAR(8)	User ID
AUTH	CHAR(1)	Authority: R=Read; W=Write; A=DIRREAD; B=DIRWRITE; X=NEWREAD; Y=NEWWRITE.

Table 54. EOCAT Record (CATTYPE = E), Record Length = 289 bytes		
Field Name	**Field Type**	**Description.**
CATTYPE	CHAR(1)	Catalog Type
OID	CHAR(8)	External Object or Link OID
SEQUENCE NUMBER	INTEGER(4)	Sequence Number
TYPE	CHAR(1)	Type of Object: X'00'=SFS external object; X'01'=BFS external link.
TYPE DEFINITION	CHAR(8)	Product Defined Type Name
DATA	CHAR(255)	External Object or Link Data
NUMBER	INTEGER(4)	Number of Entries

Table 54. EOCAT Record (CATTYPE = E), Record Length = 289 bytes (continued)		
Field Name	**Field Type**	**Description.**
RESERVED01	CHAR(4)	Reserved Area - contains 0
RESERVED02	CHAR(4)	Reserved Area - contains 0

Table 55. ACAT Record (CATTYPE = R), Record Length = 177 bytes		
Field Name	**Field Type**	**Description.**
CATTYPE	CHAR(1)	Catalog Type
OID	CHAR(8)	Unresolved Alias Object ID
DIROWNER	CHAR(8)	Directory Owner
N1	CHAR(16)	Directory Name1
N2	CHAR(16)	Directory Name2
N3	CHAR(16)	Directory Name3
N4	CHAR(16)	Directory Name4
N5	CHAR(16)	Directory Name5
N6	CHAR(16)	Directory Name6
N7	CHAR(16)	Directory Name7
N8	CHAR(16)	Directory Name8
FN	CHAR(8)	File name
FT	CHAR(8)	File type
RESERVED01	CHAR(4)	Reserved Area - contains 0
RESERVED02	CHAR(4)	Reserved Area - contains 0
RESERVED03	CHAR(8)	Reserved Area - contains 0

Table 56. NAMECAT Record (CATTYPE = N), Record Length = 88 bytes

Field Name	Field Type	Description.
CATTYPE	CHAR(1)	Catalog Type
FSID	CHAR(8)	File space ID (USEROID)
PDOID	CHAR(8)	OID of parent directory
NAMESEG	CHAR(32)	A segment of the name
NAMLN	CHAR(2)	Length of object name
OID	CHAR(8)	OID of this object
NID	INTEGER(4)	File space unique number assigned to name
INO	INTEGER(4)	File space unique number assigned to object
SCID	INTEGER(4)	Status Change Identifier
TYPE	CHAR(1)	Status of entry: 'A'=BFS regular file; 'B'=block special file; 'C'=character special file; 'D'=directory; 'X'=external link; 'P'=FIFO; 'L'=symbolic link; 'U'=unallocated row.
RESERVED01	CHAR(8)	Reserved
RESERVED02	CHAR(4)	Reserved
RESERVED03	CHAR(4)	Reserved

Table 57. NOVCAT Record (CATTYPE = V), Record Length = 57 bytes

Field Name	Field Type	Description.
CATTYPE	CHAR(1)	Catalog Type
FSID	CHAR(8)	File space ID (USEROID)
NID	INTEGER(4)	NID in corresponding NAMECAT
SEQ	INTEGER(4)	Sequence Number
NAMESEG	CHAR(32)	Nth name segment
RESERVED01	CHAR(4)	Reserved

Table 57. NOVCAT Record (CATTYPE = V), Record Length = 57 bytes (continued)		
Field Name	**Field Type**	**Description.**
RESERVED02	CHAR(4)	Reserved

Return Codes and Reason Codes

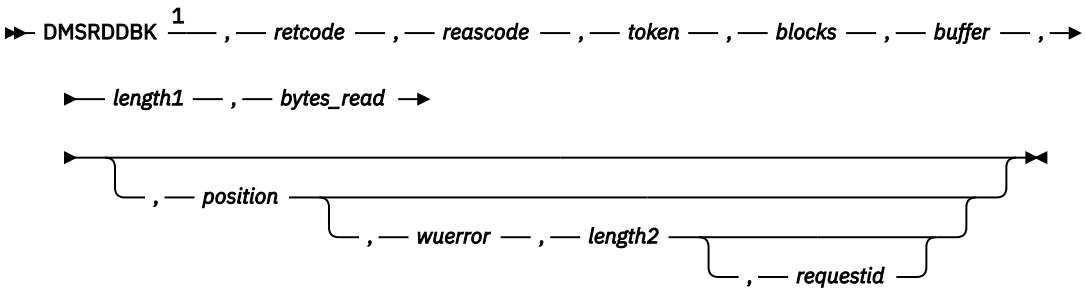
For lists of the possible return codes from DMSRDCAT, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSRDCAT. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSRDCAT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	44040	No more entries to follow.
ERROR	44900	The catalog storage group contains no data.
ERROR	56400	Catalog is not open, or is not open with intent READ, READEXT or FILEATTR.
ERROR	72100	User has attempted a Read Catalog request for a GROUP intent other than WRITE and a FILEPOOL RENAME command is currently in process on a file space in that storage group.
WARNING	78112	A file in DFSMS/VM migrated status was encountered during a READ CATALOG operation.
ERROR	90215	Invalid buffer length.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified, must be 0 or 1.
ERROR	95700	No open file pool object found for the specified token.

DMSRDDBK - Read Data Block



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for invoking a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSRDDBK, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRDDBK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRDDBK routine to get data blocks from a file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRDDBK.

reascode
(output, INT, 4) is a variable for the reason code from DMSRDDBK.

token
(input, CHAR, 8) is a variable for passing the value that uniquely identifies the file to be read. This value was returned from previous call to DMSOPDBK.

blocks
(input, INT, 4) is a variable for specifying the number of blocks to be read.

buffer
(output, CHAR, *length1*) is a variable for specifying an area into which the data is to be read.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

bytes_read
(output, INT, 4) is a variable that is set to the number of bytes of data actually placed in *buffer*. This is set to 0 if there was an error.

position
(input, INT, 4) is a variable for specifying the number of the first block to be read, relative to the beginning of the file (block 1). If *position* is not specified, or 0 is specified, the next sequential block is read.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a length field (*length2*).

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the same effect as omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and will result in an error return code.

requestid
(input/output, INT, 4) identifies an asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid* which is passed on a later Check (DMSCHECK) request.

A given DMSRDDBK request may not require communication with the file pool server. In this case, the operation is performed synchronously regardless of the value specified in *requestid*, and the value of the *requestid* parameter is not be changed by the operation.

Usage Notes

1. DMSRDDBK updates the read pointer so that, if the *position* parameter is specified as 0 (or is omitted) on the next call to DMSRDDBK, reading begins following the last block read by this call to DMSRDDBK.
2. The first read operation on a file reads block 1 unless you specify the *position* parameter to indicate a different block.
3. If DMSRDDBK is successful (the return code is 0 or 4), the requested data is returned in *buffer* and the amount of data read is returned in *bytes_read*.
4. The amount of data that is placed in the buffer is the number of blocks requested multiplied by the block size of the file. The amount of data placed in the buffer is less than this amount when:
   - The position parameter is greater than the last data block. Nothing is placed in the buffer and an end of file warning is returned.
   - The end of the file is reached before the number of blocks requested have been read. The contents of the buffer beyond the end of the file are unpredictable, because the buffer is not cleared. The number of bytes actually read into the buffer is returned in *bytes_read*.
5. The *bytes_read* parameter will contain the product of the block size and the number of blocks read following a successful read.
6. The status of the buffer area is unpredictable when the return code is 8 or 12; data in the buffer before the DMSRDDBK may have been modified.
7. Data blocks are not truncated by DMSRDDBK. If the buffer is not large enough to hold a data block, no part of that block is put in the buffer. If the buffer size is less than the data block size an error results.
8. The buffer used by DMSRDDBK should not be examined or changed while the asynchronous request is pending. Otherwise, inconsistent results may be obtained.
9. For an asynchronous request, a return code is given indicating whether the request was accepted for processing or was immediately rejected.
10. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
11. All minidisk requests are done synchronously.

Return Codes and Reason Codes

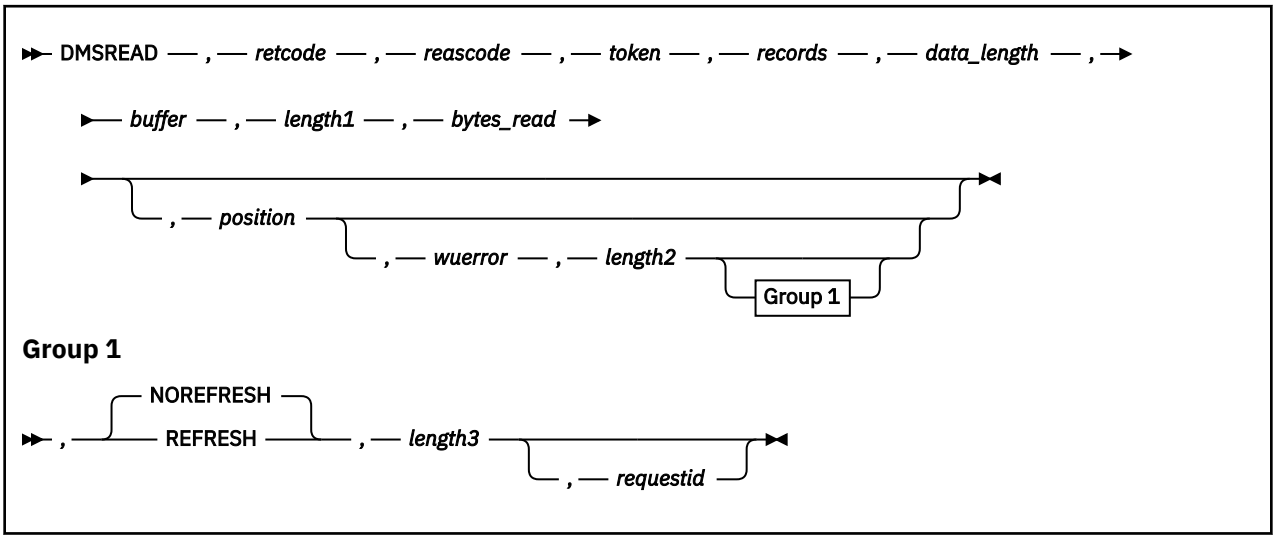
For lists of the possible return codes from DMSRDDBK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSRDDBK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSRDDBK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	51050	Call to DMSRDDBK was successful, but your file space warning threshold was reached or exceeded. This can occur on a read request because of CMS buffering.
WARNING	90101	Read was successful, but the output buffer was too small to hold all of the requested data. The data is truncated to the buffer size.
WARNING	90102	Read was successful, but the output buffer was too small to hold all of the requested data, and your file space warning threshold was reached or exceeded. The data is truncated to the buffer size.
WARNING	90103	No blocks read. End of file was reached, or the position parameter specified a record number greater than the number of blocks in the file.
ERROR	10000	File is not open using the DMSOPDBK routine with an intent of READ.
ERROR	51000	Storage group space limit exceeded.
ERROR	54000	System error. Attempt to read logical block number not associated with the file.
ERROR	90108	Size of output buffer is not greater than zero.
ERROR	90111	Invalid buffer address.
ERROR	90112	Position specifies a negative block number.
ERROR	90113	Position plus the number of blocks to read exceeds 2^{31} - 1, which is the file system capacity.
ERROR	90129	There are already 2^{31} - 1 blocks in the file.
ERROR	90270	Buffer is not large enough to hold at least one block. No data read.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	*requestid* must be 0 or 1.
ERROR	90490	Number of blocks to read is not greater than zero.
ERROR	95700	System error. No open file was found for internal token passed to SFS.
ERROR	95750	No file opened using DMSOPDBK was found for the specified token.

DMSREAD - Read



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSREAD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSREAD
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSREAD" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSREAD routine to get one or more records from a file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSREAD.

reascode
(output, INT, 4) is a variable for the reason code from DMSREAD.

token
(input, CHAR, 8) is a variable for passing the value that uniquely identifies the file to be read. This value was returned from previous call to DMSOPEN.

records
(input, INT, 4) is a variable for specifying the number of records to be read. For a file containing variable-length records, this must be a 1.

data_length
(input, INT, 4) is a variable for specifying the number of bytes of the data to be read. For a file with variable-length records, this parameter specifies the length of the record to be read. For a file with fixed-length records, this parameter must be equal to the product of the *records* parameter and the

logical record length. The value specified must not exceed the size of the buffer area specified by the *buffer* parameter. See the usage notes for additional information.

buffer
(output, CHAR, *length1*) is a variable for specifying an area into which the data is to be read. The declared length of this variable must correspond to the value you specify in the *length1* parameter.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

bytes_read
(output, INT, 4) is a variable that is set to the number of bytes of data actually placed in *buffer*. This is set to 0 if there was an error, but not if there was a warning.

position
(input, INT, 4) is a variable for specifying the number of the first record to be read, relative to the beginning of the file (record 1). If *position* is not specified, or 0 is specified, the next sequential record is read.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a corresponding length field (*length2*).

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the same effect as omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

REFRESH
(input, CHAR, 7) retrieves the most recent version of the requested data that is available to the reader. REFRESH enables readers to access data as it is written to an INPLACE file by a single concurrent writer.

It is meaningful only for SFS files with the INPLACE attribute; REFRESH is ignored for files with the NOTINPLACE attribute and for minidisk files.

NOREFRESH
(input, CHAR, 9) uses data in CMS's file system buffers where possible. This optimizes the performance of the DMSREAD operation. NOREFRESH is the default.

length3
(input, INT, 4) specifies the length of the preceding character parameter, either 7 or 9.

requestid
(input/output, INT, 4) identifies an asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid* which is passed on a later Check (DMSCHECK) request.

A given DMSREAD request may not require communication with the file pool server. In this case, the operation is performed synchronously regardless of the value specified in *requestid*, and the value of the *requestid* parameter is not be changed by the operation.

Usage Notes

1. DMSREAD updates the read pointer so that, if the *position* parameter is specified as 0 (or is omitted) on the next call to DMSREAD, reading begins following the last record read by this call to DMSREAD. Current position is not affected by a COMMIT.
2. The first read operation on a file reads record 1 unless you specify the *position* parameter to indicate a different record. You may also use the DMSPOINT function to alter the read pointer in the file.

3. Because it is permissible in files with fixed-length records to write a record with a position number more than one greater than the number of the last record, it is possible to read a record that has never been written. In this case, CMS returns a record of all X'00'.
4. Variable-length records of up to 65,535 bytes long are supported by CMS. They are read only one at a time. When reading variable-length records, a record that is longer than the buffer length is truncated. Null (zero-length) variable-length records are not supported.
5. If DMSREAD is successful (the return code is 0 or 4), the requested data is returned in *buffer* and the amount of data read is returned in *bytes_read*.
6. The status of the buffer area is unpredictable when the return code is 8 or 12; data in the buffer before the DMSREAD may have been modified.
7. An end-of-file warning (reason code 90103) is returned **only** when nothing is placed in the buffer. If an attempt is made to read more records than remain in the file and the position parameter (or its default value) is not beyond the end of the file, no warning is returned and all remaining data in the file (up to a maximum of *data_length* bytes) is placed in the buffer.
8. The *data_length* parameter specifies the maximum number of bytes to be read. The amount of data placed in *buffer* is less than *data_length* when:
   - The *position* parameter specifies a record beyond the current end of the file. In this case, nothing is placed in *buffer*, and an end-of-file warning (reason code 90103) is returned to the caller.
   - The end of the file is reached before the buffer is filled because more records were requested than remained in the file. No end-of-file warning is returned.
   - For a file with fixed-length records, the product of the *records* parameter times the logical record length is less than the value specified by *data_length*. In this case, the product is the number of bytes placed in the user's buffer, and no warning is returned to the caller.
   - The file has variable-length records and the length of the record being read is less than *data_length*.

   When the file is read successfully but the amount of data placed in *buffer* is less than *data_length*, the buffer space beyond the records read in may contain unpredictable characters because DMSREAD reads records **without** clearing the buffer first. The number of bytes actually put in the buffer is returned in *bytes_read*.
9. The *data_length* parameter should not contain a value larger than the length of the buffer. If *data_length* exceeds the buffer length, reading stops when the buffer is filled; the remainder of the current record, and perhaps subsequent records, is missed; and a truncation warning is returned.

   At the next call to DMSREAD, reading starts at the record specified in the *position* parameter. If *position* contains 0, then reading starts at the record following the last record specified on the previous request.
10. If the amount of data implied by the read request is greater than the value of *data_length*, a truncation warning (reason codes 90101 and 90102) is returned to the user. This situation occurs when:
    - The file has fixed-length records and the value of *records* times the logical record length is greater than *data_length*.
    - The file has variable-length records and the length of the record is greater than *data_length*.

    data_length should be equal to the product of the logical record length (*lrecl*) and the number of records to be read (*records*). It is not an error, however, if *data_length* is not equal to this product. When it is not, it is possible to skip data in the file during sequential reading. For example, suppose a file has twenty 80-byte fixed-length records. If the first call to DMSREAD requests five records and specifies a data length of 300, the first three records and the first 60 bytes of the fourth record are placed in the buffer and a truncation warning is returned. If the next read request does not specify a position number, reading begins with record 6, thereby skipping the last 20 bytes in record 4 and all of record 5.
11. If you are reading an INPLACE file, you may see another user's uncommitted changes to existing records without first closing and then reopening the file. (When you see updates depends on a variety

of factors.) If you need to see uncommitted updates to INPLACE files, use the FORCE option for writing and the REFRESH option for reading to control when you see the updates.

12. When you use REFRESH and NOREFRESH to read data from INPLACE SFS files:
    - Specifying REFRESH can cause considerable performance overhead. You should use it only when there is a need to read data written to the file by a concurrent writer who is using the FORCE option on DMSWRITE or is issuing frequent commit requests.
    - If the application needs to ensure that the file is subject to INPLACE updates, specify the INPLACE attribute on the DMSOPEN request. If the overwrite attribute of the file does not match the one specified on the DMSOPEN request, a warning is generated.
    - Specifying REFRESH does not enable an application to read blocks appended to the file after the reader opened the file.
    - Data in blocks that contained no data when the file was opened are not accessible to the reader. Reading records that include these blocks in an INPLACE file can result in invalid or obsolete data.
    - When a file has the INPLACE attribute and there are no concurrent writers, DMSREAD with the NOREFRESH option performs better and retrieves the same data as with the REFRESH option. If an INPLACE file has a concurrent writer, NOREFRESH may cause data to be read in an unpredictable manner and possibly at a moment when the data is inconsistent, because some or all of the data being returned to the user is being retrieved from CMS file system buffers rather than directly from SFS.
13. The buffer used by DMSREAD should be neither examined nor changed while the asynchronous request is pending. Otherwise, inconsistent results may be obtained.
14. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
15. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
16. All minidisk requests are done synchronously.

Return Codes and Reason Codes

For lists of the possible return codes from DMSREAD, see Appendix D, “Return Codes,” on page 597.

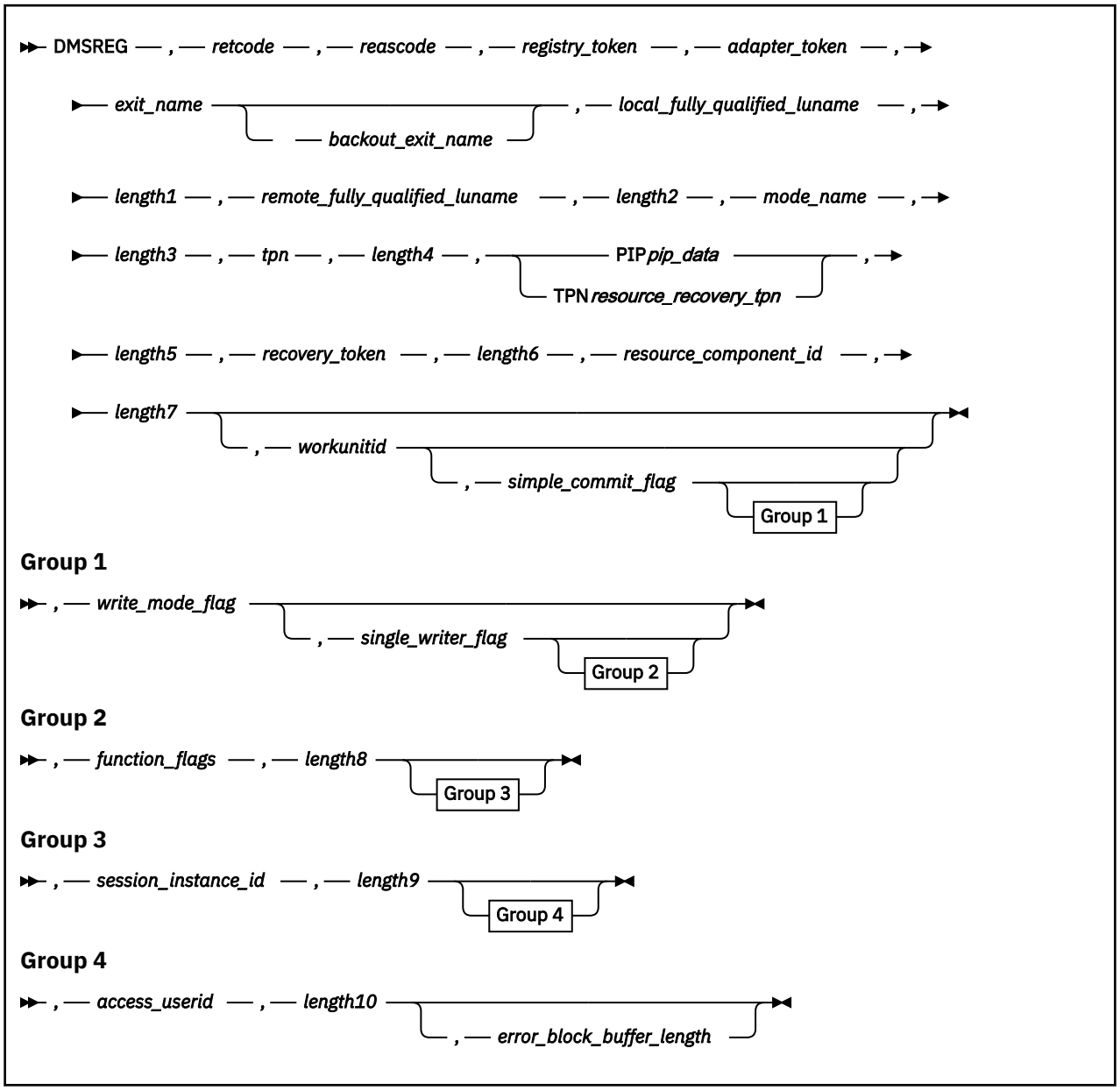
The following table lists the special reason codes returned by DMSREAD. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSREAD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	51050	Call to DMSREAD was successful, but your file space warning threshold was reached or exceeded. This can occur on a read request because of CMS buffering.
WARNING	90101	Read request was successful, but the buffer was too small to hold all of the requested data. The data is truncated to the buffer size.
WARNING	90102	Read request was successful, but the buffer was too small to hold all of the requested data, and your file space warning threshold was reached or exceeded. The data is truncated to the buffer size.

Severity	Reason Code	Description
WARNING	90103	No records read. End of file was reached, or the position parameter specified a record number greater than the number of records in the file.
ERROR	10000	System error. File is not open by DMSOPEN.
ERROR	51000	Storage group space limit exceeded.
ERROR	54000	System error. Attempt to read logical block number not associated with the file.
ERROR	90105	Invalid record format.
ERROR	90106	Number of records to read is not greater than zero.
ERROR	90107	Number of records to read is not exactly one for a file containing variable length records.
ERROR	90108	Size of buffer is not greater than zero.
ERROR	90111	Invalid buffer address.
ERROR	90112	Position specifies a negative record number.
ERROR	90113	Position plus the number of records to read exceeds 2^{31}-1, which is the file system capacity.
ERROR	90117	The variable length record read is invalid. The length is either zero or outside of the range (1 to logical record length).
ERROR	90129	There are already 2^{31}-1 blocks in the file.
ERROR	90310	Invalid option in CSL parameter list - must be REFRESH or NOREFRESH.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	*requestid* must be 0 or 1.
ERROR	95700	System error. No open file was found for internal token passed to SFS.
ERROR	95750	No file opened using DMSOPEN was found for the specified token.

DMSREG - CRR Resource Adapter Registration



Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSREG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSREG
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSREG" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSREG routine to register a resource and its adapter with the CRR synchronization point manager (SPM). The resource adapter calls this routine when there is work for the resource that requires coordination through CRR synchronization point processing, such as committing or backing out changes. Registration also supplies information needed by resynchronization processing to recover from any problems that occur during a synchronization point. As the output of DMSREG, the SPM assigns a registry token that identifies the instance of registration (resource and resource adapter pair on the work unit).

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSREG.

reascode
(output, INT, 4) is a variable for the reason code from DMSREG.

registry_token
(output, CHAR, 8) is a variable for the value that the SPM assigns to identify this instance of registration (resource and resource adapter pair on this CMS work unit). The resource adapter must save the registry token to use as an input parameter in other CRR routines.

adapter_token
(input, CHAR, 4) is a variable for the value that the resource adapter uses to identify the resource, because the resource adapter may be handling multiple resources. When driving a sync point exit to the resource adapter, the SPM passes this value in the exit routine to help the resource adapter identify the affected resource.

exit_name
(input, CHAR, 8) is a variable for the name of the CSL routine to be called by the SPM to drive an exit to this resource adapter for the various sync point functions (according to the settings of the flags in the *function_flags* parameter).

IBM does not provide this exit routine. For information about writing the routine, see *z/VM: CMS Application Development Guide*.

The name must be in uppercase. If the name is less than 8 characters, it must be left justified and padded with blanks on the right.

backout_exit_name
(input, CHAR, 8) is a variable for the name of an optional CSL routine to be called by the SPM to drive an exit to this resource adapter for the backout-required sync point function (according to the setting of the backout-required flag in the *function_flags* parameter). Resource adapters are permitted to provide a separate backout exit routine because the type of processing allowed in the backout-required exit is so restricted.

IBM does not provide this exit routine. For information about writing the routine, see *z/VM: CMS Application Development Guide*.

If used, the name must be in uppercase and must be separated from the *exit_name* parameter by a single blank. If the name is less than 8 characters, it must be left justified and padded with blanks on the right.

The length of the *exit_name backout_exit_name* field must be 17 characters. If a separate backout exit routine is not being used, the *backout_exit_name* parameter must be set to all blanks. In that case, the SPM calls the *exit_name* routine when driving an exit for the backout-required function.

local_fully_qualified_luname
(input, CHAR, 0–17) is a variable for the fully qualified LU name of the application's virtual machine (the local partner in the conversation).

If you are using CPI Communications, you can obtain this value and its length by calling the XCELFQ (Extract Local Fully Qualified LU Name) routine. For information about XCELFQ, see the *Common Programming Interface Communications Reference*, SC26-4399.

If you are using the APPC/VM assembler interface, you can obtain this value and its length from the connect complete extended data provided by the CMSIUCV CONNECT macro. For information about CMSIUCV CONNECT, see the *z/VM: CMS Macros and Functions Reference*.

If you are not using APPC to communicate with your resource manager (that is, no local fully qualified LU name exists for the conversation), specify this parameter as 0.

length1
(input, INT, 4) is a variable for specifying the length of the *local_fully_qualified_luname* parameter.

If the *local_fully_qualified_luname* parameter is 0, the *length1* parameter must also be 0.

remote_fully_qualified_luname
(input, CHAR, 0–17) is a variable for the fully qualified LU name of the resource manager (the remote partner in the conversation).

If you are using CPI Communications, you can obtain this value and its length by calling the XCERFQ (Extract Remote Fully Qualified LU Name) routine. For information about XCERFQ, see the *Common Programming Interface Communications Reference*.

If you are using the APPC/VM assembler interface, you can obtain this value and its length from the connect complete extended data provided by the CMSIUCV CONNECT macro. For information about CMSIUCV CONNECT, see the *z/VM: CMS Macros and Functions Reference*.

If you are not using APPC to communicate with your resource manager (that is, no remote fully qualified LU name exists for the conversation), specify this parameter as 0.

length2
(input, INT, 4) is a variable for specifying the length of the *remote_fully_qualified_luname* parameter.

If the *remote_fully_qualified_luname* parameter is 0, the *length2* parameter must also be 0.

mode_name
(input, CHAR, 0–8) is a variable that identifies the set of conversation characteristics, such as pacing levels and class of service. Information on the mode name is in the *Systems Network Architecture Transaction Programmer's Reference Manual for LU 6.2*, GC30-3084.

If you are using CPI Communications, you can obtain this value and its length by calling the CMEMN (Extract Mode Name) routine. For information about CMEMN, see the *Common Programming Interface Communications Reference*.

If you are using the APPC/VM assembler interface, you can obtain this value through the CMSIUCV RESOLVE macro. For information about CMSIUCV RESOLVE, see the *z/VM: CMS Macros and Functions Reference*.

If you are not using APPC to communicate with your resource manager (that is, no mode name exists for the conversation), specify this parameter as 0.

length3
(input, INT, 4) is a variable for specifying the length of the *mode_name* parameter.

If the *mode_name* parameter is 0, the *length3* parameter must also be 0.

tpn
(input, CHAR, 1–24) is a variable for the resource manager's transaction program name (TPN).

If you are using CPI Communications, you can obtain this value and its length by calling the XCETPN (Extract TP Name) routine. For information about XCETPN, see the *Common Programming Interface Communications Reference*.

If you are using the APPC/VM assembler interface, you can obtain this value and its length from the CMSIUCV RESOLVE macro. For information about CMSIUCV RESOLVE, see the *z/VM: CMS Macros and Functions Reference*.

If you are not using APPC to communicate with your resource manager, you must still supply a TPN value. In that case, the TPN is the resource ID that the resource manager uses on the CPI Communications XCIDRM (Identify Resource Manager) routine, or on the z/VM *IDENT system service, to identify the resource manager as a CRR participant. For information about XCIDRM, see the *Common Programming Interface Communications Reference*. For information about *IDENT, see *z/VM: CP Programming Services*.

length4
(input, INT, 4) is a variable for specifying the length of the *tpn* parameter.

PIP *pip_data*
(input, CHAR, 5–28) specifies Program Initialization Parameters (PIP data) to be sent concurrent with an APPC ALLOCATE (APPC/VM CONNECT) during resynchronization recovery. (Note that the actual *pip_data* length can be 1–24). The PIP keyword must be uppercase and must be separated from the *pip_data* variable by a single blank.

The CRR recovery server uses the PIP data to identify a resynchronization connection to the resource manager. If PIP data is supplied, it must include control data. This routine does not check for proper formatting of the PIP data that you provide. For more information about PIP data, see *z/VM: CP Programming Services*.

If you specify PIP data, you must omit the TPN *resource_recovery_tpn* parameter. Resynchronization will occur by using the regular TPN.

If you are using CPI Communications, you cannot specify PIP data. You must use the TPN *resource_recovery_tpn* parameter instead.

TPN *resource_recovery_tpn*
(input, CHAR, 5–28) is the resource manager's transaction program name for recovery. (Note that the actual *resource_recovery_tpn* length can be 1–24). The TPN keyword must be uppercase and must be separated from the *resource_recovery_tpn* variable by a single blank.

If this TPN is specified, the CRR recovery server uses it when allocating a conversation with the resource manager for resynchronization recovery. This enables the resource manager to recognize the conversation as one for recovery purposes, and not for the normal transmission of data.

Although usually unique, this name could be the same one used in the *tpn* parameter above. If the name is unique, the resource manager must use the CPI Communications XCIDRM (Identify Resource Manager) routine or the z/VM *IDENT system service to make this TPN (resource ID) known to the system. For information about XCIDRM, see the *Common Programming Interface Communications Reference*. For information about *IDENT, see *z/VM: CP Programming Services*.

If you specify this parameter, you must omit the PIP *pip_data* parameter.

If this parameter contains only blanks, the CRR recovery server uses the name in the *TPN* parameter to allocate a conversation with the resource manager for recovery.

length5
(input, INT, 4) is a variable for specifying the length of the PIP *pip_data* parameter or TPN *resource_recovery_tpn* parameter.

recovery_token
(input, CHAR, 0–8) is a variable for a unique value that the resource manager can assign to identify this logical unit of work for this resource adapter. During resynchronization recovery, the resource manager can use this value, if available, to identify the task requiring resynchronization. See usage note “5” on page 450.

length6
(input, INT, 4) is a variable for specifying the length of the *recovery_token* parameter.

resource_component_id
(input, CHAR, 0 or 9) is a variable for the component ID for the resource being registered. This ID, which must be specified in uppercase, is used to identify the resource type when detailed error information is returned.

The component IDs of IBM products are defined in the *Programming Systems General Information Manual*, G229-2228. Non-IBM products must define their own IDs.

length7
(input, INT, 4) is a variable for specifying the length of the *resource_component_id* parameter.

workunitid
(input, INT, 4) is a variable that identifies a group of related operations to be committed or backed out as a unit. If the work unit ID is 0 or not specified, the current CMS default work unit is used.

simple_commit_flag
(input, CHAR, 1) is a variable for indicating whether this resource supports a simple (one-phase) commit:

1
ON. Simple commit is allowed. The SPM and the CRR recovery server do not do any logging for this resource. This is the default.

0
OFF. Simple commit is not allowed. SNA LU 6.2 protocols, including logging, must always be followed for the resource.

write_mode_flag
(input, CHAR, 1) is a variable for indicating whether this resource should be treated as if there are updates to be committed or backed out:

1
ON. The resource should be treated as write-mode. A commit of the resource would leave it in a different state than a backout.

0
OFF. The resource may be treated as read-only. No recovery of the resource is needed in case of a failure during sync point processing, because from a recovery point of view a backout of the resource is equivalent to a commit. Therefore, the SPM and the CRR recovery server do not do any logging for this resource. This is the default.

single_writer_flag
(input, CHAR, 1) is a variable for indicating, when the write mode flag is set on, whether this resource is the only write-mode resource permitted for the work unit:

1
ON. This is the only write-mode resource permitted for the work unit. This is a compatibility interface provided to resource adapters communicating with resource managers that do not support the CRR two-phase commit. When registering with this flag set on, the simple commit flag should also be set on.

0
OFF. Other resources can be in write mode for the work unit at the same time as this resource. This is the default.

function_flags
(input, CHAR, 0 or 5) is a field in which the characters are positional values that indicate (left to right) whether the SPM should call an exit to this resource adapter for the following sync point functions:

- Precoordination
- Coordination
- Postcoordination
- End of work unit
- Backout required

Each character can have a value of 1 or 0:

1
ON. Call the resource adapter's exit routine for this function. For all functions except backout-required, the SPM calls the routine specified in the *exit_name* parameter. For the backout-required

function, the SPM calls the *backout_exit_name* routine, if specified. If a *backout_exit_name* routine is not specified, the SPM calls the *exit_name* routine.

0
OFF. Do not call the resource adapter's exit routine for this function.

If this parameter is not specified, the default for all positions is 1 (ON).

length8
(input, INT, 4) is a variable for specifying the length of the *function_flags* field.

session_instance_id
(input, CHAR, 0–8) is a variable for a value that the resource manager can assign to identify the conversation between the resource manager and the resource adapter for this logical unit of work. This value is meaningful only if the resource adapter and resource manager are on different systems, communicating through ACF/VTAM. During resynchronization recovery, the resource manager can use this value, if available, to identify the conversation to stop activity on it and deallocate. See usage note "5" on page 450.

If you are using CPI Communications, this value is not available.

If you are using the APPC/VM assembler interface, you can obtain the session instance ID and its length from the connect complete extended data provided by the CMSIUCV CONNECT macro. For information about CMSIUCV CONNECT, see the *z/VM: CMS Macros and Functions Reference*.

Even if the resource adapter and resource manager are not using APPC, this value may be supplied or defined by the resource manager.

length9
(input, INT, 4) is a variable for specifying the length of the *session_instance_id* parameter.

access_userid
(input, CHAR, 0–8) is a variable for the ID that the resource manager uses to determine if access to the resource should be granted to the resource adapter.

If you are using CPI Communications, you can obtain this value and its length by calling the XCECSU (Extract Conversation Security User ID) routine. For information about XCECSU, see the *Common Programming Interface Communications Reference*.

If you are using the APPC/VM assembler interface, you can obtain this value and its length from the connect complete extended data provided by the CMSIUCV CONNECT macro. For information about CMSIUCV CONNECT, see the *z/VM: CMS Macros and Functions Reference*.

If not specified, this parameter defaults to the virtual machine ID.

length10
(input, INT, 4) is a variable for specifying the length of the *access_userid* parameter.

error_block_buffer_length
(input, INT, 4) is a variable for specifying the length (in bytes) of the buffer that the SPM should reserve to hold error blocks for this registration.

If this parameter is specified, the value must be at least 4, because the first four bytes of the buffer are reserved to hold the length of the buffer. Values of 1, 2, or 3 are not allowed.

If this parameter is not specified, the default is 0, and no error block buffer is reserved. See usage note "8" on page 450.

Usage Notes

1. For guidance information on using the DMSREG routine in the context of getting a resource manager to participate in CRR, see *z/VM: CMS Application Development Guide*.
2. Parameters may not have preceding blanks. When a parameter contains multiple tokens (as in the PIP *pip_data* parameter), the tokens must be separated by single blanks.
3. If the CRR recovery server is not available, the SPM checks the *simple_commit_flag* parameter and *write_mode_flag* parameter for each resource registered on the work unit:

- If all of the resources support a simple commit, and only one resource is registered in write mode, a one-phase commit is possible for the work unit. DMSREG will be successful.
- If any resource does not support a simple commit, or more than one resource is registered in write mode, a two-phase commit is required. Because the CRR recovery server is needed to do the necessary logging, DMSREG will fail.

If the resource adapter calls DMSREG and gets a return code stating the registration failed because the CRR recovery server is not available, the resource adapter must not allow any updates to be made to the resource that require coordination through CRR.

4. The CSL routines identified by the *exit_name* parameter and *backout_exit_name* parameter must be loaded before DMSREG is called. These routines must not be dropped after they have been identified to the SPM on a DMSREG call, unless the resource adapter unregisters the resource with a call to DMSUNREG.

   IBM does not provide either of these exit routines. For information about writing the routines, see *z/VM: CMS Application Development Guide*.
5. In a transaction with two partners, the partners by definition share the same LUWID value. If both partners access the same resource, then it is possible for both partners to have tasks for the resource manager that require resynchronization processing. If resynchronization recovery is necessary, the LUWID alone cannot distinguish between the tasks. The CRR recovery server passes the recovery token and session instance ID, if available, to the resource manager in the Compare-States request. The resource manager can use the session instance ID or a combination of the recovery token and the LUWID to identify the task.
6. If the simple commit flag is set off, CRR uses the two-phase protocol, including logging, regardless of the setting of the single writer flag.
7. The resource adapter should always leave the end-of-work-unit function flag set on in the *function_flags* parameter, so the resource adapter can be notified to clean up and unregister when the work unit completes. Clean-up might involve such things as releasing storage and severing paths. Resource adapters must unregister as part of their end-of-work-unit processing.
8. If you register with the *error_block_buffer_length* parameter omitted or set to 0, a 4-byte default buffer is passed to the resource adapter when its exit is driven. However, because the first four bytes of the buffer are used to specify the buffer's length, the default buffer has no room for any error data.

Return Codes and Reason Codes

For lists of the possible return codes from DMSREG, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSREG. WARNING means the request was processed, but a warning condition was encountered; ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8.

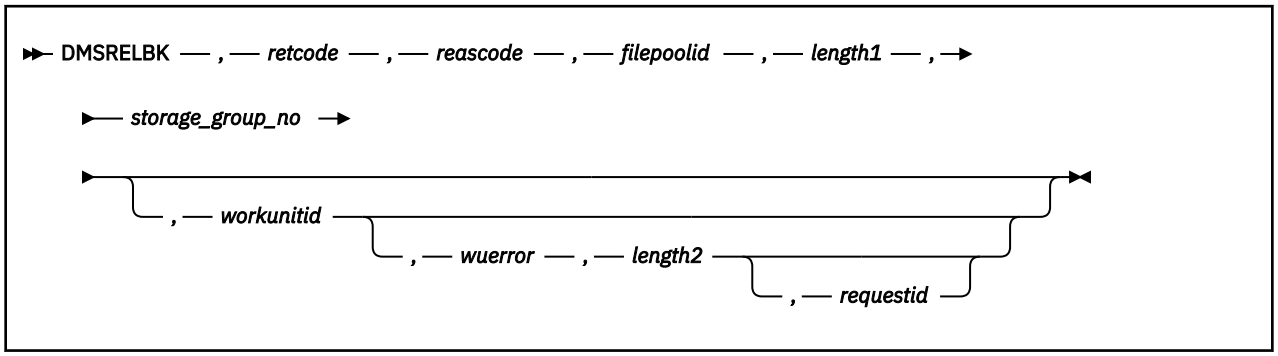
DMSREG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	79053	The CRR recovery server is not available, so the resource has been registered as a single writer.
ERROR	55000	Insufficient virtual storage in the CRR recovery server.
ERROR	75000	The service levels of the CRR recovery server and the user machine are not compatible.
ERROR	78001	The *exit_name* parameter is incorrect or is not followed by a blank.
ERROR	78002	The *tpn* parameter is not 0–24 bytes in length.

Severity	Reason Code	Description
ERROR	78005	The *local_fully_qualified_luname* parameter is not 0–17 bytes in length.
ERROR	78006	The *remote_fully_qualified_luname* parameter is not 0–17 bytes in length.
ERROR	78007	The *mode_name* parameter is not 0–8 bytes in length.
ERROR	78009	The keyword preceding the *pip_data* variable or *resource_recovery_tpn* variable is not either PIP or TPN.
ERROR	78010	The PIP *pip_data* parameter is not 5–28 bytes in length.
ERROR	78011	The TPN *resource_recovery_tpn* parameter is not 5–28 bytes in length.
ERROR	78012	The *recovery_token* parameter is not 0–8 bytes in length.
ERROR	78013	The *resource_component_id* parameter is not 0 or 9 bytes in length.
ERROR	78014	The *simple_commit_flag* parameter is not 0 or 1.
ERROR	78015	The *write_mode_flag* parameter is not 0 or 1.
ERROR	78016	The *single_writer_flag* parameter is not 0 or 1.
ERROR	78017	The *function_flags* parameter does not contain five characters, or contains a character other than 0 or 1.
ERROR	78018	The *session_instance_id* parameter is not 0–8 characters long.
ERROR	78019	The *access_userid* parameter is not 0–8 characters long.
ERROR	78024	Incorrect *error_block_buffer_length* parameter.
ERROR	79051	The routine identified by the *exit_name* parameter or *backout_exit_name* parameter is not loaded.
ERROR	79052	The CSL template for the exit routine defined an incorrect number of parameters.
ERROR	79054	The CRR recovery server is not available but is needed to do a two-phase commit of the work unit.
ERROR	79055	The *write_mode_flag* parameter was set on for this resource, but a different resource is already in write mode for the work unit with the *single_writer_flag* parameter set on.
ERROR	79056	The *write_mode_flag* parameter and *single_writer_flag* parameter were set on for this resource, but a different resource is already in write mode for the work unit.
ERROR	79057	The work unit is ending.
ERROR	79058	A sync point is in progress for the work unit.
ERROR	81050	System error. An unexpected return code was received from a call to DMSCSL fast path initialization for the *exit_name* routine or *backout_exit_name* routine.
ERROR	90540	Incorrect *workunitid* parameter.
ERROR	95200	System error. Further attempts to access the CRR recovery server will be rejected.

Severity	Reason Code	Description
ERROR	97280	Your attempt exceeds the number of APPC/VM connections allowed for the CRR recovery server.

DMSRELBK - Release Blocks



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSRELBK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRELBK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRELBK routine to release alternate blocks in the specified storage group that had encountered an I/O error when attempting to write user data. File pool administration authority is required.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRELBK.

reascode
(output, INT, 4) is a variable for the reason code from DMSRELBK.

filepoolid
(input, CHAR, 1-8) is a variable that identifies the file pool containing the storage group affected. When specifying this parameter, an appended colon is invalid and should not be used.

length1
(input, INT, 4) is a variable containing the length of the preceding character parameter (*filepoolid*).

storage_group_no
(input, INT, 4) contains a number between 2 and 32767, and not greater than the MAXDISKS parameter used on the FILESERV GENERATE command, which identifies the storage group affected.

workunitid
(input, INT, 4) is a variable identifying the work unit associated with the DMSRELBK routine. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. DMSRELBK is an atomic request. This means that there can be no outstanding work for the file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSRELBK causes the work in the file pool to be committed without coordination.
2. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
3. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

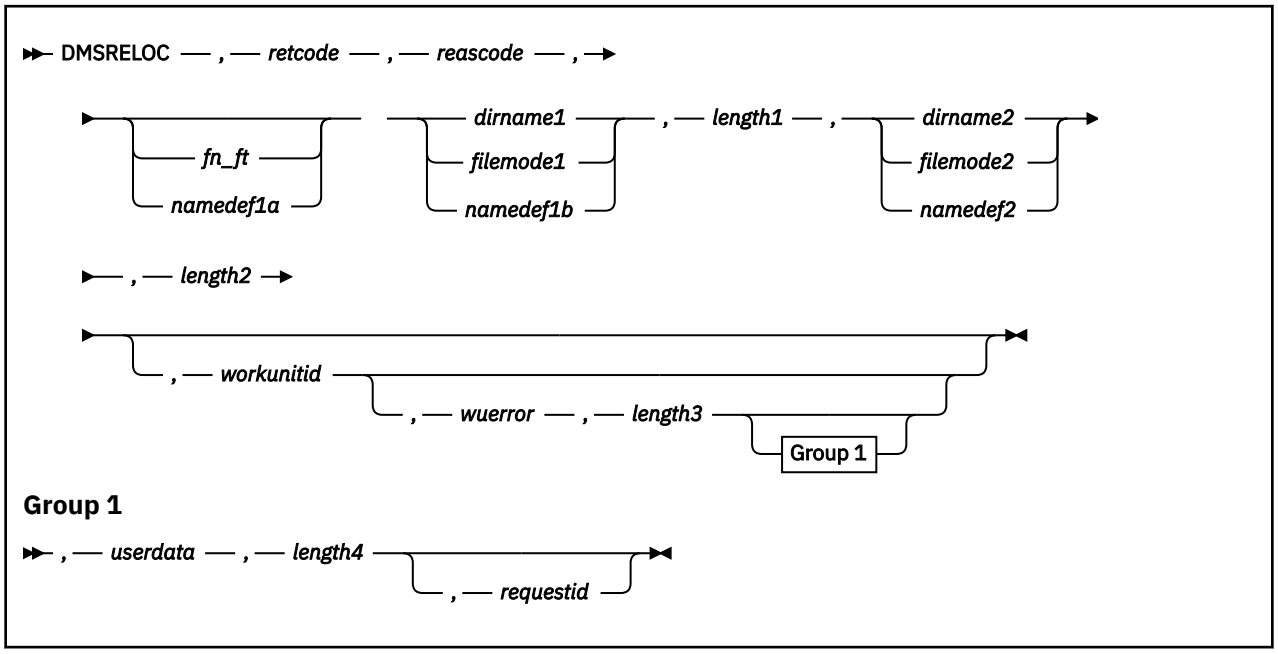
For lists of the possible return codes from DMSRELBK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSRELBK. ERROR indicates that the request failed. Errors cause return codes 8 or 12.

DMSRELBK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	30000	You do not have administrator authority.
ERROR	50100	Specified storage group number is invalid (less than 2 or greater than MAXDISKS server parameter).
ERROR	50200	Specified storage group does not exist.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid requestid specified, must be 0 or 1.
ERROR	90476	Invalid file pool ID specified.
ERROR	90540	Invalid work unit ID specified.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

DMSRELOC - Relocate



Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSRELOC is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRELOC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRELOC routine to move a file, alias, external object, or directory subtree from one SFS directory to another.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRELOC.

reascode
(output, INT, 4) is a variable for the reason code from DMSRELOC.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file to be moved. It can specify a base file, alias, or external object.

namedef1a
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *fn_ft*.

dirname1
(input, CHAR, 1-153) is a variable for specifying the directory name.

To relocate a file, specify *fn_ft*. Use the *dirname1* parameter to specify the name of the directory that contains the file to be moved. Files cannot be relocated from directory control directories. If you specify *fn_ft* or *namedef1a* to identify a file, the file must reside in a file control directory.

To relocate a directory subtree, do not specify *fn_ft*. The name you specify in the *dirname1* parameter should be the top node of the directory subtree that is to be moved (this cannot be your top directory). All subdirectories and files will be moved. The directory subtree name specified in *dirname1* will no longer exist. Directories moved by specifying a directory subtree can be either file control or directory control.

filemode1
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef1b*) rather than a file mode. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef1b
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *dirname1* or *filemode1*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1a*, if specified; plus *dirname1*, *filemode1*, or *namedef1b*). See "Compound Variables" on page 15 for coding details.

dirname2
(input, CHAR, 1-153) is a variable for specifying the directory name.

To relocate a file, specify *fn_ft*. Use the *dirname2* parameter to specify the name of the directory to which the file is being moved (this cannot be the same as that specified in *dirname1*). You cannot relocate files into a directory control directory.

To relocate a directory subtree, do not specify *fn_ft*. Use the *dirname2* parameter to specify the name of the new parent directory for *dirname1*. The name specified in *dirname2* cannot be the parent or a subdirectory of *dirname1*. Directory subtrees can be relocated into either directory control or file control directories.

filemode2
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *dirname2* or *filemode2*.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*dirname2*, *filemode2*, or *namedef2*).

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify

a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable containing the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. Relocating a directory subtree does not change the authorities to and aliases of the objects within it.
2. A relocated external object takes on the authorizations granted to the new parent directory.
3. Users with NEWREAD or NEWWRITE authority to a directory are granted appropriate READ or WRITE authority to a file that is relocated into the directory.
4. A relocated alias remains in effect.
5. The issuer of the Relocate routine must be the owner of the file or the directory. DMSRELOC works only within a single file pool and a single user's directory structure.
6. Access to a directory is not broken by DMSRELOC. A file control directory can be relocated while other users have the directory accessed.

   A directory control directory, on the other hand, cannot be relocated if another user has the directory accessed read/write; in that case, your attempt to relocate the directory will fail. Users who have the directory accessed in read-only mode will not learn of the change until they reaccess the directory. If you have the directory control directory accessed, it must be accessed in read/write status.
7. If you attempt to relocate a directory subtree in such a way that the result would be a directory with more than eight qualifiers, DMSRELOC fails.
8. If a file is open when you attempt to relocate it, DMSRELOC fails.
9. If you call DMSRELOC for a nonrecoverable file and then do a rollback, the file is still relocated.
10. When a file is open in the directory being relocated, or in a subdirectory, DMSRELOC fails.
11. If a file or directory is locked, it cannot be relocated except when the issuer has an UPDATE or EXCLUSIVE lock on the file or directory. In this case, the file or directory can be relocated.
12. DMSRELOC is an atomic request. This means that there can be no outstanding work for the affected file pool on the specified work unit (or the default work unit if none was specified) when this routine is called. When it is finished, DMSRELOC will cause a noncoordinated commit to be done for the work in the affected file pool.
13. You cannot relocate files, aliases, and external objects out of or into a directory control directory. You can, however, relocate a directory structure whose parent is a directory control directory. You can also relocate file control or directory control directories into a directory control directory. In either case, if you have the parent directory control directory accessed, you must have it accessed read/write.
14. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the

data. The ESM might, for example, require you to specify a password for the file or directory you are relocating.

15. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
16. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSRELOC, see Appendix D, "Return Codes," on page 597.

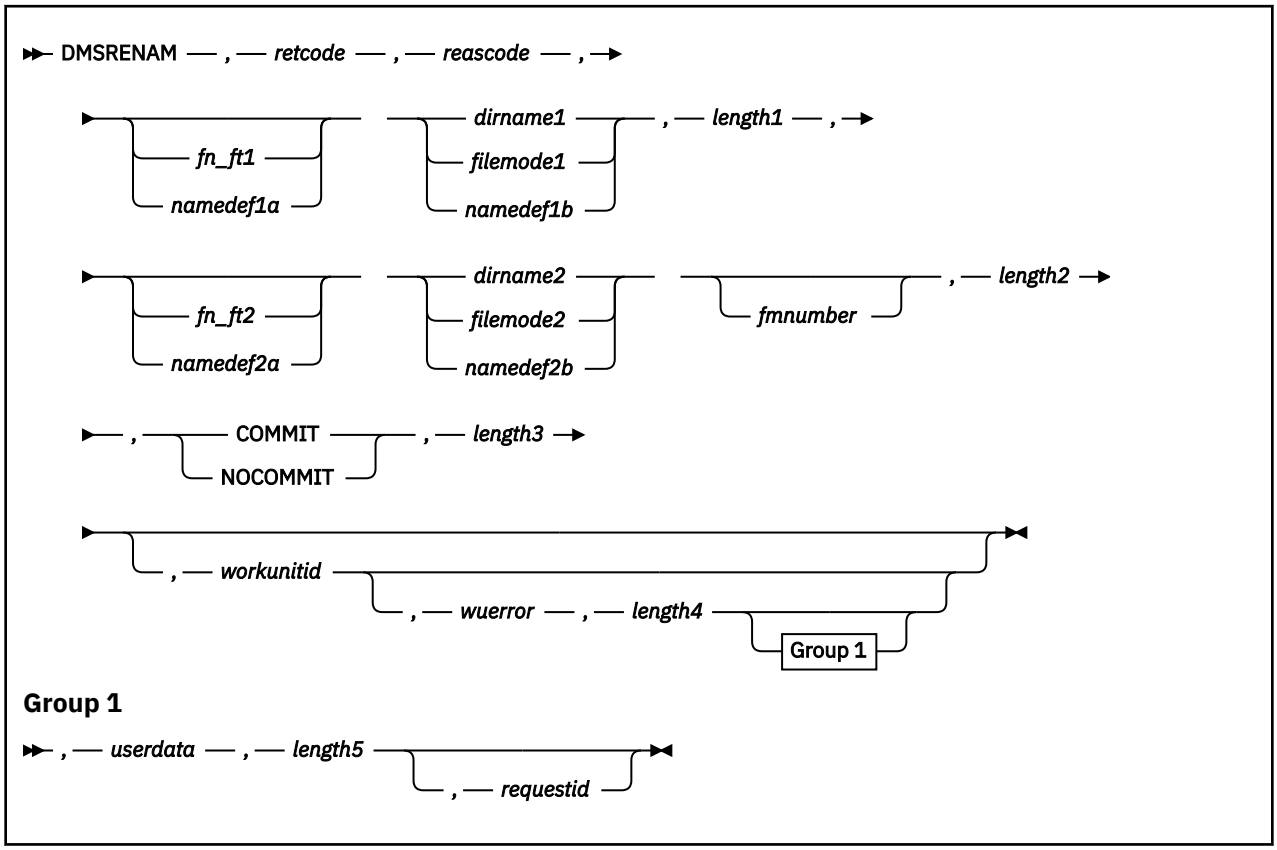
The following table lists the special reason codes returned by DMSRELOC. ERROR indicates that the request failed. Errors cause return code 8 or 12.

DMSRELOC can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	20000	The directory to be relocated already exists in the target directory.
ERROR	20010	The file to be relocated already exists in the target directory.
ERROR	44000	The directory or file to be relocated does not exist or you are not authorized to relocate it.
ERROR	44100	The target directory does not exist or you are not authorized to write to it.
ERROR	60200	You have attempted to relocate a top-level directory.
ERROR	61000	Your Relocate request would have resulted in a directory name having more than eight levels.
ERROR	61200	The parent of the directory to be relocated has the same name as the specified target directory.
ERROR	61300	The directory to be relocated has the same name as the specified target directory.
ERROR	61400	You have attempted to relocate a directory such that the directory would be a subdirectory of itself.
ERROR	61500	The directory to be relocated and the target directory are not in the same directory hierarchy.
ERROR	63700	Specified directory control directory or its parent is accessed read-only.
ERROR	63800	Specified directory is a directory control directory.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90300	Incorrect parameter in CSL parameter list. File mode number erroneously specified, or file name and file type is specified in the target *dirname*.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Severity	Reason Code	Description
ERROR	90410	Incorrect length specified for one of the character variables.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90460	File pool IDs in source and target directory names are not the same.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and file pool ID was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.
ERROR	90611	Input file mode letter did not represent an accessed SFS directory in the second file ID.
ERROR	95400	This work unit was already active for the specified file pool when DMSRELOC was executed.

DMSRENAM - Rename



Context

File System Management: SFS and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSRENAM is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRENAM
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRENAM routine to rename an SFS file, alias, external object, or subdirectory, or a minidisk file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRENAM.

reascode
(output, INT, 4) is a variable for the reason code from DMSRENAM.

fn_ft1
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file, alias, or external object to be renamed. The issuer of the call must be the owner of the subdirectory or have the minidisk file mode accessed read/write.

namedef1a
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *fn_ft1*.

dirname1
(input, CHAR, 1-153) is a variable for specifying either the name of the directory containing the file or the subdirectory to be renamed. If *fn_ft1* or *namedef1a* is specified, use the *dirname1* parameter to specify the name of the directory the file is in. If *fn_ft1* or *namedef1a* is **not** specified, use *dirname1* to specify the subdirectory whose last qualifier is to be renamed. The issuer of the call must be the owner of the subdirectory.

filemode1
(input, CHAR, 1) is a variable for specifying the file mode of the file or SFS subdirectory to be renamed. If the character in this field is also a namedef it is treated as a namedef rather than as a file mode letter.

namedef1b
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *dirname1* or *filemode1*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft1* or *namedef1a*, if specified; plus *dirname1*, *filemode1*, or *namedef1b*). See "Compound Variables" on page 15 for coding details.

fn_ft2
(input, CHAR, 3-17) is a variable for specifying the new file name and file type of the file.

namedef2a
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *fn_ft2*.

dirname2
(input, CHAR, 1-153) is a variable for specifying either the directory of the renamed file or the renamed subdirectory. If *fn_ft2* or *namedef2a* is specified, this is the name of the directory or file mode containing the renamed file. It must be the same as *dirname1*. If *fn_ft2* or *namedef2a* is **not** specified, this is the new name of the subdirectory. The name specified in the *dirname2* parameter **cannot** be the same as *dirname1*.

filemode2
(input, CHAR, 1) is a variable for specifying the file mode of the file or directory. If the character in this field is a one character namedef, it is treated as a namedef rather than as a file mode letter. The file mode must be the same as *filemode1* or resolve to the same SFS directory specified in *dirname1*.

namedef2b
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *dirname2* or *filemode2*.

fmnumber
(input, CHAR, 1) is a variable for specifying the file mode number. If omitted, it remains the same. If you are renaming an alias, the file mode number must be the same as the base file's.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft2* or *namedef2a*, if specified; plus *dirname2 filemode2* or *namedef2b*; plus *fmnumber*). See "Compound Variables" on page 15 for coding details.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes. You cannot specify the NOCOMMIT parameter when renaming a directory. If the file resides on a minidisk, this will result in the minidisk master file directory not being written to the minidisk. This performs the same function as the NOUPDIRT option on the CMS Rename command.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (COMMIT or NOCOMMIT).

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an SFS external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length5* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. All authorities and aliases remain the same.
2. The issuer of the DMSRENAM for a directory must be the owner of the subdirectory.
3. You cannot rename a file or an external object to a different directory, subdirectory or minidisk.
4. The new name of the file, external object, or directory cannot exist when DMSRENAM is issued.
5. Only one subdirectory level can be renamed at a time.
6. When the subdirectory is renamed, all subdirectories below it are also renamed for that part of their name.
7. When renaming an alias of a base file, only the alias is renamed. DMSRENAM cannot be used to change the file mode number of an alias.
8. If you call DMSRENAM for a nonrecoverable file and then do a rollback, the file is still renamed.
9. Users accessing a directory do not lose their access when the directory is renamed. A subsequent QUERY SEARCH or a QUERY ACCESSED will show the new directory name.
10. You cannot rename a file in an SFS directory if the file is open in read/write mode by any user. You can rename a file that you or another user has open in read-only mode. You also cannot rename a file, if it is already open, using the FSOPEN, DMSOPEN, or DMSOPDBK interfaces.

11. You cannot rename a directory control directory, or a file, external object, or subdirectory of a directory control directory, while the parent directory is accessed read/write by another user or read-only by you. You can rename it while it is accessed read-only by other users; until they reaccess the directory, the QUERY SEARCH and QUERY ACCESSED commands will return the old name.
12. If any files or subdirectories within a directory are open, the directory cannot be renamed.
13. If a file is locked, it cannot be renamed except when the issuer has an UPDATE or EXCLUSIVE lock on the file. In that case, the file can be renamed.
14. If any files or subdirectories within a directory to be renamed are locked, the directory cannot be renamed, except when the issuer has an UPDATE or EXCLUSIVE lock on the file or subdirectory. In that case, the directory can be renamed.
15. Renaming directories without committing the work is not supported. The NOCOMMIT parameter is not allowed when using DMSRENAM to rename a directory.
16. You can rename a file in another user's file control directory if you have write authority to both the file and the directory. You can rename an external object in another user's file control directory if you have write authority to the directory. You can also rename another user's alias if you have read authority to the base file and write authority to the directory.

    You can rename another user's file in a directory control directory if you have directory write authority for the directory.
17. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the file or directory you are renaming. If the directory being opened is minidisk, *userdata* is not used.
18. If the COMMIT parameter is specified and the return code is 8, then either:
    - An error occurred during the processing of the Rename operation, or
    - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
19. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
20. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
21. For a minidisk file, when the file is renamed, the minidisk directory is updated (unless NOCOMMIT is specified). If COMMIT was specified, the work unit will be committed. If an error occurs attempting to commit the work unit the operations associated with that work unit will remain uncommitted but the minidisk file will be renamed.
22. Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Return Codes and Reason Codes

For lists of the possible return codes from DMSRENAM, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSRENAM. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

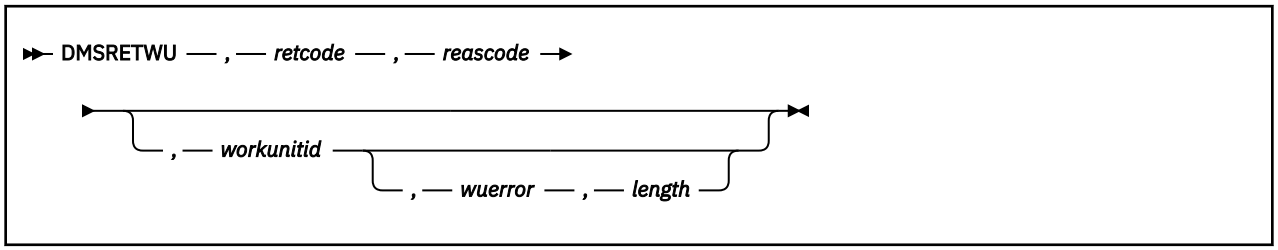
Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

DMSRENAM can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	The specified 'new' directory name already exists in the specified file pool.
ERROR	20010	The file IDs are identical. The source file or directory name is the same as the target.
ERROR	44000	The directory, alias, external object, or file to be renamed does not exist or you are not authorized to rename it.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	60100	You have the directory or file open.
ERROR	60200	You have attempted to rename a top-level directory.
ERROR	61600	Attempt to rename other than the lowest level of a directory name.
ERROR	61610	Invalid file mode change.
ERROR	61700	Attempt to rename a file to a directory or a directory to a file.
ERROR	61800	Attempt was made to change file mode number of an alias. The file mode number of the alias remains the same as the file mode number of the base file.
ERROR	61900	You cannot change file mode numbers when renaming aliases.
ERROR	63700	Specified directory control directory is accessed read-only.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90300	Invalid parameter in CSL parameter list. It must be COMMIT or NOCOMMIT. COMMIT is required for subdirectories, and this reason code is returned if NOCOMMIT is specified.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Severity	Reason Code	Description
ERROR	90410	Invalid length specified for one of the character variables.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.
ERROR	90430	The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.
ERROR	90440	The specified file mode number is invalid. It must be a numeral between 0 and 6.
ERROR	90450	Global characters (* or %) were found in either the file name or file type.
ERROR	90460	File pool IDs in source and target directory names are not the same.
ERROR	90472	Invalid request ID specified; must be 0 or 1.
ERROR	90500	The specified directory name is invalid.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is invalid.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.
ERROR	90601	Attempt to rename an SFS directory but file mode letter corresponds to a minidisk.
ERROR	90614	Rename attempted on a read/only minidisk file.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSRETWU - Return Work Unit ID



Context

Work Unit Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSRETWU is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSRETWU
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSRETWU routine to tell the system that your application has completed all work on the specified work unit, and that the system can free its resources associated with that work unit. Before your application calls DMSRETWU, it should close all SFS files and directories that were open under the work unit, and commit all work. Valuable error information can be lost if DMSRETWU performs those functions.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSRETWU.

reascode
(output, INT, 4) is a variable for the reason code from DMSRETWU.

workunitid
(input, INT, 4) is a variable you can use to specify the work unit to be returned. If not specified or 0, the current default work unit is returned.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning extended SFS error information for file pools involved in the work unit. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

Usage Notes

1. SFS files and directories that are open under this work unit are closed.

2. An attempt is made to commit any outstanding work on the work unit. If the commit fails, outstanding work on the work unit is rolled back.
3. The work unit stack is not checked or modified by this routine.
4. Work units that were associated with a user ID, security token, or user accounting data when the work unit IDs were assigned by DMSGETWU are no longer associated with these items after they have been returned.
5. All protected conversations associated with this work unit are deallocated.
6. DMSRETWU does not delete event monitors created by CMS multitasking applications. For information on using asynchronous requests in CMS multitasking applications, see the *z/VM: CMS Application Development Guide*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSRETWU, see Appendix D, "Return Codes," on page 597.

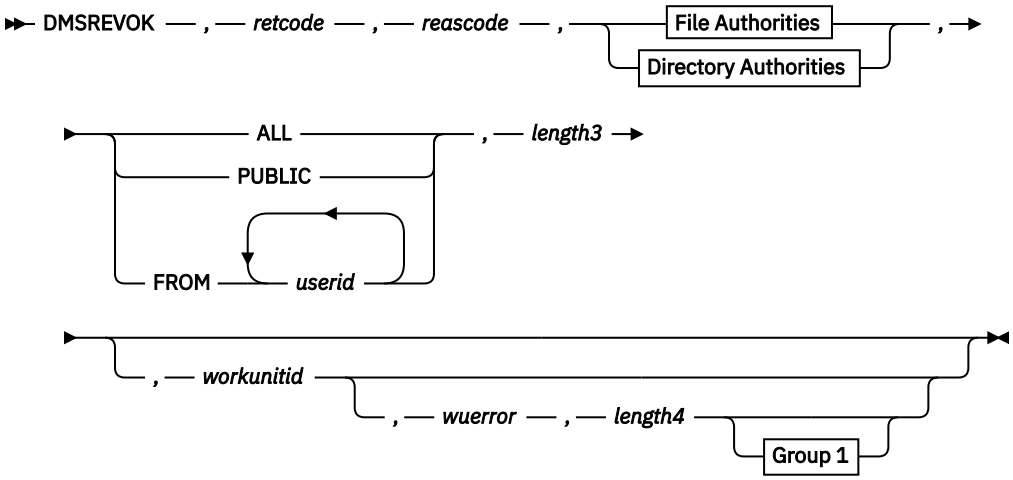
The following table lists the special reason codes returned by DMSRETWU. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. SEVERE ERROR means that the state of different resources may be inconsistent. Warnings cause return code 4, errors cause return code 8 or 12, and severe errors cause return code 16 or 20.

DMSRETWU can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

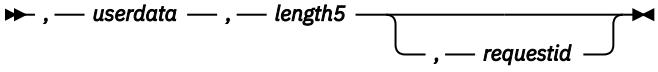
Severity	Reason Code	Description
WARNING	51060	The user exceeded the SFS file space threshold for one or more file spaces.
WARNING	76070	Changes to non-recoverable files have been lost. This code is returned in the *error_reascode* field of FPERROR. The FPERROR information is returned only if a data area is specified with the *wuerror* parameter.
WARNING	81054	Resynchronization is in progress on one or more protected resources and the consistency state is unknown. In other words, if the resynchronization is successful, all resources are committed.
WARNING	81055	All protected resources have committed. However, your processing environment has changed or the availability of one or more resources has changed. For example, some protected conversations may have been deallocated or some open files may have been closed.
WARNING	96802	One or more nonrecoverable files could not be closed.
ERROR	35000	The work unit was rolled back. System limits were exceeded when attempting to commit. CMS cannot commit more than approximately 230 resources on a work unit.
ERROR	54500	The CRR recovery server log limits were exceeded.
ERROR	79058	A synchronization point is already in progress for the work unit. This request was ignored.
ERROR	81053	The work unit was returned successfully, but outstanding work was rolled back.

Severity	Reason Code	Description
ERROR	81054	The commit operation failed. A resynchronization operation is in progress to finish rolling back one or more protected resources. Their consistency state is unknown. If the resynchronization is successful, all protected resources will be rolled back.
ERROR	81056	The CRR recovery server is not available. The work unit was rolled back; all protected resources have been restored to their prior mutually consistent state.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90540	Invalid work unit ID. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSRETWU.
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement. This is an SFS-specific reason code.
ERROR	96100	Insufficient virtual storage. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSRETWU. This request was ignored.
SEVERE ERROR	81051	A protocol violation may have provoked an inconsistent state among the resources. **RC=16** The commit or rollback operation was completed. One or more protected resources may have rolled back. **RC=20** The commit or rollback operation failed. The work unit was rolled back, but one or more protected resources may have committed.
SEVERE ERROR	81052	Resource manager or operator intervention resulted in an inconsistent state among the resources. **RC=16** The commit operation was completed. Protected resources have been rolled back. **RC=20** The commit operation failed. During an attempt to roll back changes, protected resources were committed.

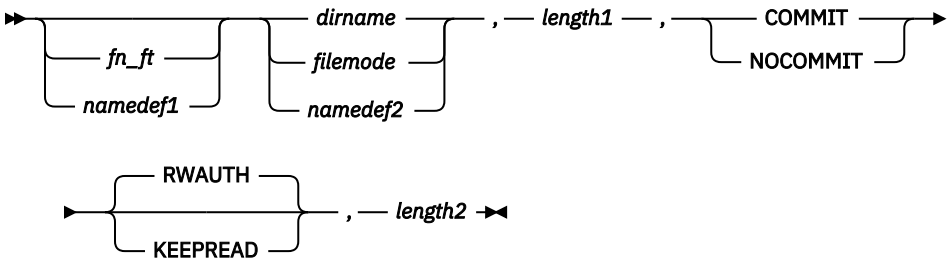
DMSREVOK - Revoke Authority



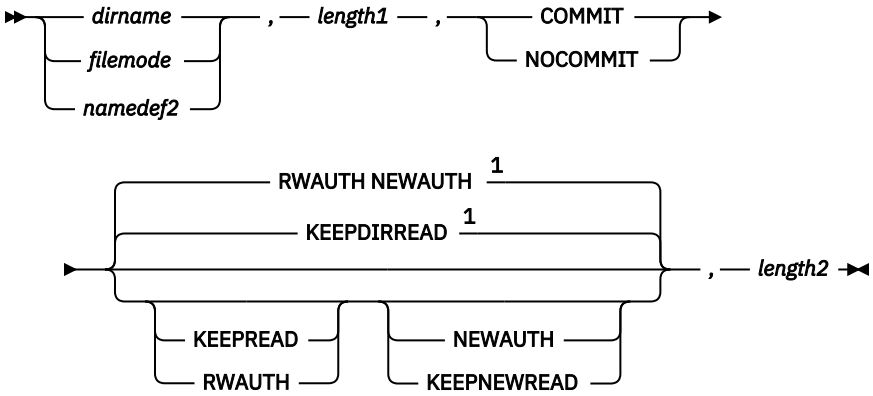
Group 1



File Authorities



Directory Authorities



Notes:

[1] RWAUTH NEWAUTH is the default for file control directories. KEEPDIRREAD is the default for directory control directories.

Context

File System Management: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSREVOK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSREVOK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSREVOK routine to revoke other users' access privileges to your SFS file or directory. It does not alter authorities held by an ESM.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSREVOK.

reascode
(output, INT, 4) is a variable for the reason code from DMSREVOK.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and the file type (separated by at least 1 blank) of the file for which authority is to be removed.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the directory name. If *fn_ft* or *namedef1* is specified, this is the directory that contains the file on which authority is to be revoked. If *fn_ft* or *namedef1* is not specified, this is the directory on which authority is being removed.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If this file mode letter is also a one character namedef (*namedef2*), the namedef will be used. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for *dirname* or *filemode*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, if specified; plus *dirname*, *filemode*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

COMMIT
(input, CHAR, 6) means keep all changes associated with the work unit, from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not keep the changes.

KEEPDIRREAD
(input, CHAR, 11) specifies that you want to change a user's authority from DIRWRITE to DIRREAD. KEEPDIRREAD is valid only when a directory is identified (without a file name and file type) and when that directory has the directory control attribute.

KEEPNEWREAD
(input, CHAR, 11) specifies that you want to change a user's authority from NEWWRITE to NEWREAD. KEEPNEWREAD is valid only when a directory is identified (without a file name and file type) and when that directory has the file control attribute.

KEEPREAD
(input, CHAR, 8) specifies that you want to change a user's authority from WRITE to READ. This parameter can only be specified for a file control directory or a file within a file control directory.

NEWAUTH
(input, CHAR, 7) specifies that you want to remove NEWREAD or NEWWRITE authority from a specified user. NEWAUTH is valid only when a directory is identified (without a file name and file type) and when that directory has the file control attribute.

RWAUTH
(input, CHAR, 6) specify this parameter when you want to remove both READ and WRITE authority from a specified user. RWAUTH is valid only for file control directories and for files residing in file control directories.

length2
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (COMMIT or NOCOMMIT, and KEEPDIRREAD, KEEPNEWREAD, KEEPREAD, NEWAUTH, or RWAUTH). See "Compound Variables" on page 15 for coding details.

ALL
(input, CHAR, 3) indicates revoke authority from all users that have been granted authority on the file or directory. If authority has been granted to PUBLIC, ALL revokes that authority also.

PUBLIC
(input, CHAR, 6) revokes PUBLIC authority. (Public authority means all users were authorized to access the file.) Revoking PUBLIC authority does not revoke authorities that were granted on an individual basis.

FROM *userid*
(input, CHAR, *length3*) indicates the user ID (or list of user IDs separated by blanks) from which authority is being revoked. Nicknames are not allowed.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (ALL, or PUBLIC, or FROM *userid*), including blanks.

workunitid
(input, INT, 4) is a variable for identifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length4*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The format and meaning of the data is defined by the ESM (see the Usage Notes).

length5
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length5* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. The authorities you can grant and revoke are:
   - For SFS file control directories:

     READ
     WRITE (which implies READ)
     NEWREAD
     NEWWRITE (which implies NEWREAD)

     NEWREAD and NEWWRITE are independent of READ and WRITE.
   - For files within file control directories:

     READ
     WRITE (which implies READ)
   - For SFS directory control directories:

     DIRREAD
     DIRWRITE (which implies DIRREAD)
   - For files within directory control directories:

     Not applicable. The ability to read from or write to files in directory control directories is derived from DIRWRITE and DIRREAD authority on the directory in which the file resides.
   - External objects

     Not applicable. An external objects have no authority associated with it. The remote name in an external object can be queried by anyone with read authority to the directory.
2. Users may not revoke authority from themselves.
3. To revoke all authorization for an object, execute DMSREVOK **without** specifying any authority parameters (KEEPREAD, KEEPNEWREAD, KEEPDIRREAD, NEWAUTH, or RWAUTH).

   If you omit the authority parameters for a file control directory, the specified user loses any READ, WRITE, NEWREAD, or NEWWRITE authority that he holds. For directory control directories, the user loses DIRREAD or DIRWRITE authority (whichever is held).

   When a file is identified and the authority parameters are omitted, the user loses READ and WRITE authority (if held). (The directory in which the file resides would have to be a file control directory because authorities cannot be granted to or revoked from individual files within directory control directories.) Specifying RWAUTH in this case has the same effect as omitting the authority parameters. That is, both READ and WRITE authority to the file are revoked.
4. To downgrade a user's authorization, specify one of the authority options. When you specify an authority option, only the indicated authority is changed. Other authorizations are retained.
5. For file control directories, you can specify two authority parameters in a single DMSREVOK call. The allowed combinations are (order is not significant):
   - NEWAUTH KEEPREAD
   - RWAUTH KEEPNEWREAD
   - KEEPREAD KEEPNEWREAD
   - RWAUTH NEWAUTH (default)
6. If a user is granted authority on a file, and authority is also granted to PUBLIC, revoking authority from PUBLIC does not revoke the authority initially granted to the user.

7. This table shows the circumstances when a file owner can and cannot revoke authority on a locked file using DMSREVOK.

Lock Creator	Lock Mode on the File		
	Explicit Share	Explicit Update	Explicit Exclusive
File Owner	No	Yes	Yes
Another User	No	No	No

8. For file control directories, authority to a file can be revoked from a user that has the file open. The revocation takes effect when the file is closed.
9. Authority for a directory cannot be revoked if there are files open within that directory.
10. You cannot revoke authority for a directory control directory if it is accessed in read/write mode (using the CMS ACCESS command) by anyone other than you. You can revoke the authority of a user accessing the directory in read-only mode, but the revocation does not take effect until the directory is released.

    If you have the directory accessed, it must be accessed in read/write status.
11. If your installation does not use an external security manager (ESM), *userdata* is not used.

    If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify the *userdata* parameter. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to supply a password for the specified file or directory.
12. If you call DMSREVOK with the *userdata* parameter for an ESM-protected file, but you do not have the proper authority on the file or specify an incorrect value in *userdata*, the routine fails and returns error reason code 44000.
13. If you call DMSREVOK without the *userdata* parameter (because the ESM does not require it) for an ESM-protected file, the routine succeeds, but warning reason code 44010 is returned.
14. If the COMMIT parameter is specified and the return code is 8, then either:
    - An error occurred during the processing of the Revoke Authority operation, or
    - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.
15. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
16. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSREVOK, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSREVOK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Note: Other return and reason codes can be returned by this routine when the COMMIT parameter is specified. See the description of the DMSCOMM (Commit) CSL routine for other possible codes.

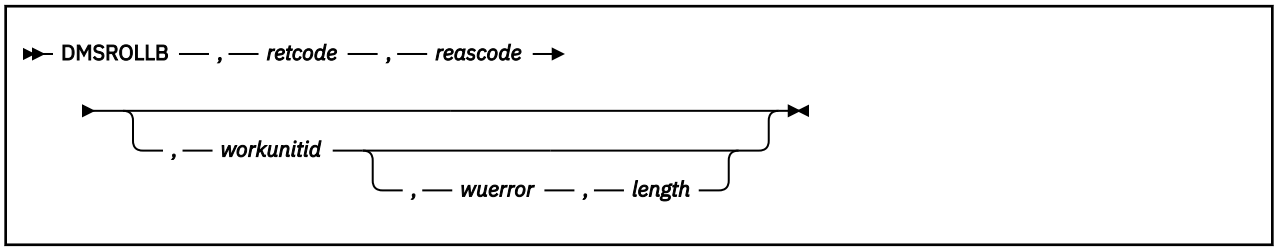
DMSREVOK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	32610	One or more specified user IDs have neither READ/WRITE nor NEWREAD/NEWWRITE authority on the object, but a REVOKE AUTHORITY was requested that implies that these authorities existed.
WARNING	32620	One or more specified user IDs does not have NEWREAD or NEWWRITE authority on the object.
WARNING	32630	One or more specified user IDs does not have DIRREAD or DIRWRITE authority on the object.
WARNING	32640	One or more specified user IDs does not have READ or WRITE authority to the specified file pool object.
WARNING	32680	One or more specified user IDs is your userid. You cannot revoke authority from yourself.
WARNING	44010	Object is protected by external security manager (revoke successful).
WARNING	51060	File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.
ERROR	10000	System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.
ERROR	10100	System error. COMMIT was specified. Conflicting file attributes.
ERROR	20000	Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.
ERROR	44000	Specified file or parent directory does not exist or you are not authorized to revoke authorities granted on the file.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	51000	COMMIT was specified. Storage group space limit exceeded.
ERROR	51100	System error. COMMIT was specified. No minidisks assigned to the storage group.
ERROR	63400	KEEPREAD, KEEPNEWREAD, NEWAUTH, RWAUTH, or *fn_ft* specified for a directory control directory.
ERROR	63500	KEEPDIRREAD specified on a file control directory.
ERROR	63600	KEEPDIRREAD, KEEPNEWREAD, or NEWAUTH specified on a base file or alias.
ERROR	63700	Specified directory control directory is accessed read-only.

Severity	Reason Code	Description
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT parameter was specified.
ERROR	90129	COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.
ERROR	90230	Specified directory does not exist or you are not authorized to revoke authorities granted on it.
ERROR	90310	Incorrect option in CSL parameter list.
ERROR	90315	Missing option in CSL parameter list.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than three tokens in the string.
ERROR	90410	Incorrect parameter length specified.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90450	Incorrect characters (* or %) were found in either the file name or file type.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	90500	The specified directory name is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.

Severity	Reason Code	Description
ERROR	95600	The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.
ERROR	95700	System error. COMMIT was specified. No open file found for internal token passed to SFS.

DMSROLLB - Rollback



Context

Coordinated Resource Recovery: SFS

Call Format

The format for calling a CSL routine is language dependent. DMSROLLB is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSROLLB
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSROLLB routine to undo changes made to one or more recoverable files or protected resources. All uncommitted work is discarded. All SFS and BFS files and directories left open on the work unit are closed.

DMSROLLB takes advantage of Coordinated Resource Recovery (CRR) to coordinate the rollback of multiple protected resources. For more information on CRR, see the *z/VM: CMS Application Development Guide*.

Note: Changes to minidisk files cannot be explicitly rolled back.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSROLLB.

reascode
(output, INT, 4) is a variable for the reason code from DMSROLLB.

workunitid
(input, INT, 4) is a variable you can use to specify the work unit to be rolled back. If you want to specify an optional parameter following *workunitid* without specifying a particular work unit, specify a value of 0 for the *workunitid* parameter. When you do this, the current default work unit is rolled back.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for returning extended error information for the file pools involved in the rollback. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of SFS extended error information that may be passed

back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

Usage Notes

1. You can use the CSL routine Get Synchronization Point Errors (DMSGETSP) to get more detailed information about errors that occurred during the rollback. If you are using a protected conversation, error data may be created even for return code 0. To get this error information, use the routine "DMSPCAER - Protected Conversation Adapter Errors" on page 355.
2. If return code 8 is encountered, your application can try to correct the problem indicated by the reason code and then reissue DMSROLLB. If it cannot correct the problem, it should abend to ensure that a rollback occurs.
3. Changes to NORECOVER files will not be rolled back as the result of an application-initiated rollback request. A rollback generally causes files with this attribute to be committed.

Return Codes

The return codes used by DMSROLLB are:

Return Code
Meaning

0
Roll back was successful.

4
Rollback was successful, with a warning condition.

8
Rollback was unsuccessful. Work is not rolled back.

12
Rollback was successful; however, the rollback was caused by an event such as a problem with one of the protected resources.

20
Rollback was successful, but one or more protected resources may have committed changes.

Return Codes and Reason Codes

For lists of the possible return codes from DMSROLLB, see Appendix D, "Return Codes," on page 597.

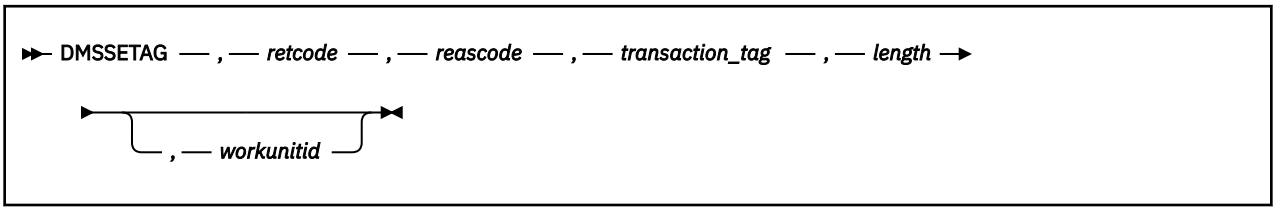
The following table lists the special reason codes returned by DMSROLLB. WARNING means the request was processed, or the desired state already exists. ERROR indicates that the request failed. SEVERE ERROR means that the state of different resources may be inconsistent. Warnings cause return code 4, errors cause return code 8 or 12, and severe errors cause return code 20.

DMSROLLB can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51060	The user exceeded the SFS file space threshold for one or more file spaces. (This occurred when closing nonrecoverable files.)
WARNING	76070	Changes to SFS nonrecoverable files have been lost. This code appears only as part of the work unit error information that is provided when you specify the *wuerror* parameter, or when you request SFS error data using the DMSGETSP (Get Synchronization Point Errors) or DMSGETER (Get My Errors) CSL routines.

Severity	Reason Code	Description
WARNING	81055	The rollback was successful, but your processing environment has been changed. For example, some protected conversations may have been deallocated, or updates to nonrecoverable SFS files may have been lost. To get resource-specific error information, use the DMSGETSP (Get Synchronization Point Errors) CSL routine; or, for SFS errors, use the contents of the *wuerror* parameter.
WARNING	96802	One or more nonrecoverable files could not be closed.
ERROR	79058	A synchronization point is already in progress for the work unit. This request was ignored.
ERROR	79061	A resource reported that the application left it in a state that does not allow a rollback, so this request was ignored and the work unit was not rolled back or committed. All protected resources are in the same state they were before the call to DMSROLLB. Use the Get Synchronization Point Errors CSL routine (DMSGETSP) to find out which protected resource caused the error. If the error occurred on a protected conversation (one allocated with a sync level of sync point), the application must deallocate the conversation. A protected conversation can cause a rollback to fail when an application: • Has allocated a protected conversation, but its partner has not accepted • Has sent data on a protected conversation, but its partner has not received the data • Issued a Receive on a protected conversation with a nonzero buffer length, but has not received all of the data yet.
ERROR	81053	The work unit was rolled back. All protected resources have been restored to their prior mutually consistent state. The rollback did not occur because of normal processing, but was caused by an occurrence such as a problem with one of the protected resources.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90540	Invalid work unit ID. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSROLLB.
ERROR	96100	Insufficient virtual storage. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSROLLB. This request was ignored.
SEVERE ERROR	81051	When trying to roll back changes, one or more protected resources may have committed due to a protocol violation.

DMSSETAG - Set Transaction Tag



Context

Coordinated Resource Recovery

Call Format

The format for calling a CSL routine is language dependent. DMSSETAG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSETAG
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSETAG routine to tell CMS to store some information on the CRR recovery server log. The local CRR recovery server operator and resource manager operators can use this information to determine what they need to do during resynchronization. Each transaction (or logical unit of work) can have its own information (tag) saved on the log at each synchronization point.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSETAG.

reascode
(output, INT, 4) is a variable for the reason code from DMSSETAG.

transaction_tag
(input, CHAR, 1-80) is a variable for specifying up to 80 characters of information about the transaction being processed. This information should enable the recovery operator (local CRR recovery server operator or local resource manager operator) to determine what needs to be done in case of a problem requiring operator intervention. The information supplied will be displayed by the CRR recovery server's CRR QUERY LU or QUERY LUWID command. SFS also supports transaction tags; the QUERY PREPARED command displays the information provided.

Transaction tags are also displayed in some resynchronization messages.

length
(input, INT, 4) is a variable for specifying the length, up to 80 bytes, of the preceding character parameter (*transaction_tag*).

workunitid
(input, INT, 4) is a variable you can use to specify the work unit this transaction tag is to be associated with. If not specified or 0, the current default work unit is used.

Usage Notes

1. You can change the transaction tag as often as necessary. Use a different tag for each synchronization point if the recovery requirements for each synchronization point are different.
2. Transaction tags are written to the local CRR recovery server logs and Shared File System (SFS) logs. Other resource managers may log the transaction tag as well. Remote CRR recovery server logs do not automatically log the transaction tag; the APPC partner application would have to receive the transaction tag in a data record or in PIP data (because it cannot get the original transaction tag by way of architected interfaces) and then call DMSSETAG with that transaction tag. For more information on using transaction tags, see the *z/VM: CMS Application Development Guide*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSSETAG, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSSETAG. ERROR indicates that the request failed. Errors cause return code 8.

DMSSETAG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	90300	Invalid input parameter in CSL parameter list. The specified transaction tag length is invalid.
ERROR	90540	Invalid work unit ID.

DMSSETR - CRR Set Received

Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSSETR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSETR
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSETR" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSETR routine to tell the CRR synchronization point manager (SPM) that a backout is required for the work unit. The resource adapter should call this routine when it receives (on a verb *outside of* synchronization point processing) an indication that a backout or session outage (resource failure) occurred for one of its resources. The SPM then passes the information to other resources on the work unit.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSETR.

reascode
(output, INT, 4) is a variable for the reason code from DMSSETR.

registry_token
(input, CHAR, 8) is a variable that identifies the affected resource. This value must have been previously returned by a successful call to DMSREG, or unpredictable results could occur.

action
(input, INT, 4) is a variable for indicating whether the resource had a backout or a failure:

4
Backout

8
Resource failure

Usage Notes

1. For guidance information on using the DMSSETR routine in the context of getting a resource manager to participate in CRR, see the *z/VM: CMS Application Development Guide*.
2. Do not call DMSSETR while the SPM is processing a sync point for the work unit.
3. DMSSETR puts the work unit in a state that does not permit it to be committed. A subsequent attempt to commit will result in a backout of the work unit.

4. The SPM will not drive a resource adapter exit for processing required by a backout if it previously received an indication of a backout or failure for that resource. (The calling resource may have already backed out.)
5. The backout of the work unit does not occur until the application issues the backout request.
6. Using this routine with an action value of 4 (backout) makes the resource adapter an initiator of the resulting backout. In other words, the resource adapter will receive the ADAIOKBO (Initiator OK Backout) action from the SPM when all agents have responded ADAOKBO (OK Backout). Using this routine with an action value of 8 (resource failure) makes the resource adapter an agent in the resulting backout.

Return Codes and Reason Codes

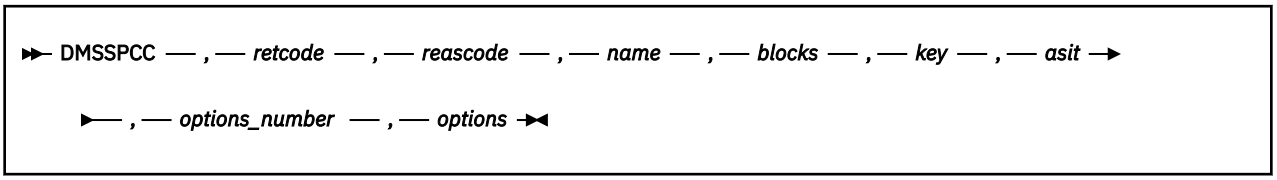
For lists of the possible return codes from DMSSETR, see Appendix D, “Return Codes,” on page 597.

The follow table lists the special reason codes returned by DMSSETR. WARNING means the request was processed, but a warning condition was encountered. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8.

DMSSETR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	79059	An action value of backout or resource failure was given, but the work unit is already in a state that requires a backout.
ERROR	78003	Incorrect *registry_token* parameter.
ERROR	78004	Incorrect *action* parameter.
ERROR	79058	The call to DMSSETR is not allowed because the SPM is already processing a sync point.
ERROR	81057	System error. The resource is not registered for the work unit.

DMSSPCC - Create Data Space



Call Format

The format for calling a CSL routine is language dependent. DMSSPCC is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCC
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPCC" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCC routine to create a data space or to make your virtual machine's primary address space shareable.

For information on using address spaces and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCC.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCC.

name
(input, CHAR, 24) is a variable used to specify the name of the data space. The name is a string of 1 to 24 characters. Only uppercase letters (A through Z), numbers (0 through 9), and certain special characters (# $ @ _ -) are allowed. If the name contains incorrect characters, then no new data space is created. The name must be padded on the right with blanks if it is not 24 characters long. The name cannot begin with DMS, because this is a preassigned CMS prefix.

If you call this routine from an XA virtual machine, you must specify BASE for this parameter.

Specifying BASE indicates that CMS structures are to be created to make your virtual machine's primary address space shareable with other virtual machines that are permitted access by your application.

When you specify BASE for this parameter, the *blocks* and *key* parameters are ignored, along with *options Set 4* and *Set 5*.

blocks
(input, INT, 4) is a variable used to specify the number of 4096-byte pages (or blocks) in the data space. The minimum size of a data space is one page (4096 bytes) and the maximum is 524,288 pages (2 gigabytes). However, the actual data space that is created is always a multiple of 256 pages. If the value specified is not a multiple of 256, it is rounded up to a multiple of 256.

key
(input, INT, 4) is a variable used to specify the storage key in which the data space should be allocated. The value specified for this parameter must be 0 to 15, representing the access key

desired. This parameter is used only if the *options Set 4* value 23 (OTHERS) is specified. Otherwise, it is ignored.

asit
(output, CHAR, 8) is a variable used for returning the address space identification token (ASIT), the unique system-wide identifier for the requested data space.

options_number
(input, INT, 4) is a variable used to specify the number of entries in the array of option values.

options
(input, INT, 4) is the first of an array of variables that contain option values. (In REXX, this is the stem of a compound variable, *without the period*.)

The option values specify the attributes to be associated with the data space that is being created. There are five sets of option values. Only one option value can be specified for each set. If multiple values are specified, the appropriate number is placed in successive entries in the array. The options parameter is required; if there is no array of options, set the value of the options number to zero, so the placeholder parameter will be ignored.

The valid option sets for this routine are:

Set 1
defines whether the data space will survive end-of-command clean up. Valid values are:

1 (NOKEEP)
specifies that the data space will be deleted during end-of-command processing. This is the default value for this option set.

2 (KEEP)
specifies that the data space will not be deleted during end-of-command processing.

Set 2
defines whether the data space will survive abnormal termination clean up. Valid values are:

10 (NOSYS)
specifies that the data space will be deleted during CMS abend processing. This is the default value for this option set.

11 (SYSTEM)
specifies that the data space will not be deleted during CMS abend processing.

Set 3
defines whether the data space can be accessed by programs in other virtual machines. Valid values are:

15 (NOSHARE)
specifies that the data space cannot be accessed by programs running in other virtual machines. This is the default value for this option set.

16 (SHARE)
specifies that the data space can be accessed by programs running in virtual machines other than the owner of the data space.

Set 4
specifies the storage key of the data space. Valid values are:

20 (USER)
specifies that the data space is to be created in user key. This is the default value for this option set.

21 (NUCLEUS)
specifies that the data space is to be created in nucleus key.

23 (OTHERS)
specifies that the protection key value to be used for the data space being created on this call is in the *key* parameter.

Set 5
defines whether the data space is to be fetch protected. Valid values are:

25 (NOFPROT)
specifies that the data space is not to be fetch protected. This is the default value for this option set.

26 (FPROT)
specifies that the data space is to be fetch protected.

Usage Notes

1. The requesting program should save the ASIT returned from the create request. This value is required as input for most of the other address space service routines.
2. If the data space is deleted and recreated, a new ASIT is issued.
3. For detailed information on the create service, see the ADRSPACE CREATE macro description in *z/VM: CP Programming Services*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCC, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by DMSSPCC. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85004	The specified data space name matches the name of a previously created data space. The data space was not created. The previously created data space was not created using DMSSPCC.
ERROR	85008	Creating the data space would cause the maximum number of allowed address spaces for your virtual machine to be exceeded. The data space was not created.
ERROR	85012	Creating the data space would cause the maximum total data space size for your virtual machine to be exceeded. The data space was not created.
ERROR	85016	The specified data space name contains incorrect characters. The data space was not created.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	85102	Incorrect parameter or options specified. This could be caused by: • An incorrect value • Two options having a duplicate value • One option's value conflicting with another's.
ERROR	85104	Illegal request for a CMS data space service. For example, the requesting virtual machine tried to create a data space with a name that already exists due to a previous call to DMSSPCC or the specified data space name begins with the preassigned CMS prefix, DMS.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPCD - Delete Data Space

Call Format

The format for calling a CSL routine is language dependent. DMSSPCD is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCD
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPCD" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCD routine to delete a data space. Only the owner of a data space can delete it.

This routine causes any access list entries of your virtual machine designating the deleted data space to be placed in **unused** state. In addition, if the data space being deleted is designated in the access list of any other permitted virtual machines, those access list entries are placed in **revoked** state. Any subsequent reference to this data space causes an addressing-capability exception.

When DMSSPCD is called from an XA virtual machine, it deletes all data space structures related to your primary address space.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCD.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCD.

asit
(input, CHAR, 8) is a variable used for specifying the address space identification token, the unique system-wide identifier for the data space to be deleted. If you call this routine from an XA virtual machine, you must specify the ASIT that identifies the virtual machine's primary address space. This indicates that all data space structures related to your primary address space are to be deleted; your primary address space is not deleted.

Usage Notes

1. After a data space is deleted, previously permitted users (applications) must call the Remove Address Space Addressability (DMSSPLR) routine to place their associated host access list entries in unused state. Once in the unused state, the host access list entry can be re-used by the Establish Addressability (DMSSPLA) CSL routine to reference another address space.
2. For detailed information on the delete service, see the ADRSPACE DESTROY macro description in *z/VM: CP Programming Services*.

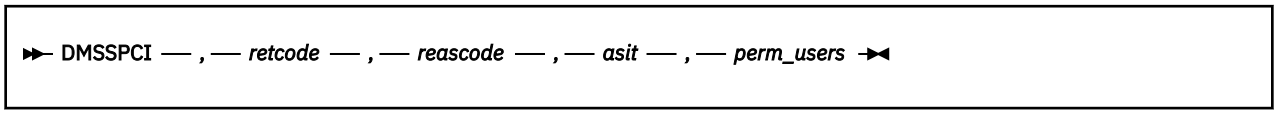
Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCD, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPCD. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85004	The specified address space identification token (ASIT) does not identify a data space that was created by your virtual machine. This could occur if a data space is not created, or if it is created using DMSSPCC, but then is deleted using a CP macro before calling DMSSPCD.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	85104	Illegal request for a data space service. For example, DMSSPCC was not called to create the data space specified for deletion on this call.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPCI - Isolate Address Space



Call Format

The format for calling a CSL routine is language dependent. DMSSPCI is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCI
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPCI" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCI routine to return an address space owned by your virtual machine to a private state. This isolates the address space, preventing access by any other virtual machine regardless of the attributes specified on the DMSSPCC (Create Data Space) routine.

If the address space to be isolated is designated in the access list of any other virtual machine, those access list entries are placed in revoked state. If another virtual machine attempts to use a revoked access list entry, an addressing-capability exception occurs.

This routine runs until all virtual machines have completed any in-progress storage accesses to the address space and all affected access list entries are set to revoked state.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCI.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCI.

asit
(input, CHAR, 8) is a variable used for specifying the address space identification token, the unique system-wide identifier for the address space to be isolated.

perm_users
(input, INT, 4) is a variable used for specifying whether the previously permitted users are to be purged from the CMS-maintained data space structures when the address space is isolated. Valid values for this parameter are:

60 (NOPURGE)
Indicates that the previously permitted users are not to be purged. These users can be repermitted access to the address space by calling the Restore Address Space Access (DMSSPCR) routine when the address space is ready to be made shareable again.

61 (PURGE)
indicates that users previously permitted should be purged. These users would then have to be explicitly granted access to the address space by calling the Permit Address Space Access (DMSSPCP) routine when the address space is ready to be made shareable again.

Usage Notes

1. This routine can require a significant amount of time and processing to complete. Its use should be minimized.
2. When specified with PURGE, DMSSPCI deletes all CMS-maintained structures representing virtual machines that were previously permitted access to the address space by the Permit Address Space Access (DMSSPCP) routine. This allows an application to allow a different group of users to access the address space when the address space is ready to be made shareable again. The new group of users would be defined by the DMSSPCP routine.

   When specified with NOPURGE, DMSSPCI leaves all CMS-maintained structures representing virtual machines that were previously permitted access to the address space by the DMSSPCP routine. This allows an application to restore access to the address space to the same group of previously permitted users when the address space is ready to be made shareable again. Use the Restore Address Space Access (DMSSPCR) routine to allow these users to access the address space again.
3. For detailed information on the isolate function, see the ADRSPACE ISOLATE macro description in *z/VM: CP Programming Services*.

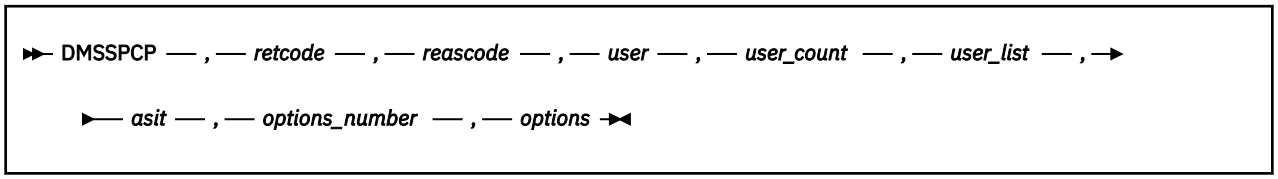
Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCI, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPCI. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Severity	Reason Code	Description
WARNING	85105	A call sequence warning occurred. DMSSPCI was called for an address space that was not previously specified on a Permit (DMSSPCP) or Restore (DMSSPCR). Either DMSSPCI was called immediately after the address space was created or it was called immediately after a prior DMSSPCI call. The address space was already in a private state, so this call had no effect.
ERROR	85004	The specified address space identification token (ASIT) does not identify a data space that was created by your virtual machine. This could occur if a data space is not created, or if it is created using DMSSPCC, but then is deleted using a CP macro before calling DMSSPCD.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	85102	Incorrect parameter or options specified. This could be provoked by: • An incorrect value • Two options having a duplicate value • One option's value conflicting with another's.
ERROR	85104	Illegal request for a CMS address space service. For example, DMSSPCC was not called to create the data space specified for isolation on this call, or the ASIT is incorrect.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPCP - Permit Address Space Access



Call Format

The format for calling a CSL routine is language dependent. DMSSPCP is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCP
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPCP" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCP routine to permit another user to access the address space identified by the address space identification token you specify. Only the owner of the address space can call this routine.

The address space to which you are permitting access must have been created with the SHARE attribute.

The authorization granted by this service persists until one of the following occurs:

- Your virtual machine subsequently converts the address space for which the user was permitted access back to the private state by using the Isolate Address Space (DMSSPCI) routine.
- Your virtual machine subsequently deletes the data space for which the user was permitted access by calling the Delete Data Space (DMSSPCD) routine.
- Your virtual machine is involved in a virtual machine reset, by using such CP commands such as SYSTEM CLEAR, SYSTEM RESET, IPL or LOGOFF.
- A virtual machine reset is performed by the user virtual machine that was permitted access to the data space.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

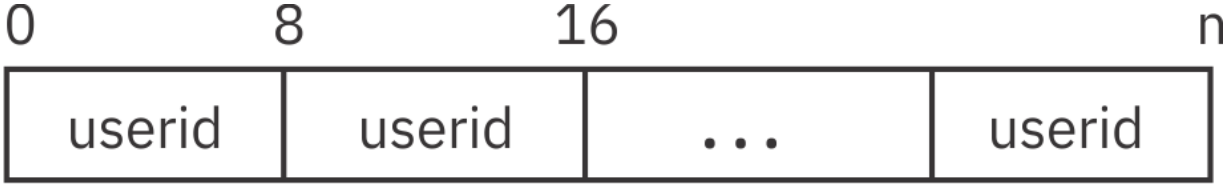
retcode
(output, INT, 4) is a variable for the return code from DMSSPCP.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCP.

user
(input, CHAR, 8) is a variable you can use to identify the virtual machine that is to be permitted to access your address space. You can specify either a user ID or a virtual configuration identification token (VCIT) in this variable. The option value 5 (USERID) indicates that this variable contains the user ID of the virtual machine that is being permitted access to the address space. The option value 6 (VCIT) indicates that the variable contains the VCIT to identify the virtual machine that is to be permitted to access the address space. This parameter must be supplied, but will be ignored if 7 (USERID LIST) is specified.

user_count
(input, INT, 4) is a variable you can use to specify the number of user IDs in the list whose address is specified in the *user_list* parameter. This must be a value between 1 and 4095. This parameter must be supplied, but is ignored if 5 (USERID) or 6 (VCIT) is specified.

user_list
(input, INT, 4) is a variable you can use to specify the address of the list of user IDs to be permitted to access the address space. This parameter must be supplied, but it is ignored if 5 (USERID) or 6 (VCIT) is specified in the *options* parameter. The user list itself is a contiguous area containing the 8-byte user IDs to be permitted access to the address space (numbers are shown in decimal):



asit
(input, CHAR, 8) is a variable used for specifying the unique system-wide identifier for the address space to which another user is being permitted access.

options_number
(input, INT, 4) is a variable used to specify the number of entries in the array of option values.

options
(input, INT, 4) is the first of an array of variables that contain option values. (In REXX, this is the stem of a compound variable, *without the period*.)

Option values specify the identity of the virtual machine being permitted to access the address space and whether this access is to include READ or READ/WRITE authority. There are two sets of option values. Only one option value can be specified for each set. If several values are specified, the appropriate number is placed in successive entries in the array. The options parameter is required; if there is no array of options, set the value of the options number to zero, so the placeholder parameter will be ignored.

The valid option sets for this routine are:

Set 1
defines whether the address space can be modified by a program running in another user's virtual machine. If LIST is specified in *Set 2*, then the access specified for this option set applies to each user ID in the list. Valid values are:

0 (READ)
specifies that the permitted program or user can only read from the address space identified by the *asit* parameter. This is the default value for this option set.

Your virtual machine is implicitly authorized for read-write access to its primary address space and all address spaces that it creates. This authorization cannot be changed for your own virtual machine by specifying the READ value in the *options* parameter.

1 (WRITE)
specifies that the permitted program or user can write to the address space identified by the *asit* parameter.

Set 2
specifies whether the identifier specified is the user ID or the VCIT of the user's virtual machine, or whether a list of user IDs is being specified. Valid values are:

5 (USERID)
specifies that the identifier specified for the *user* parameter is the user ID of the virtual machine. This is the default value for this option set.

6 (VCIT)
specifies that the identifier specified for the *user* parameter is the virtual configuration identification token.

7 (USERID LIST)
specifies that the user IDs supplied reside at the address indicated by the *user_list* parameter. Each identifier in the list is 8 bytes long. The number of user IDs in the list is specified in the *list_count* parameter.

Usage Notes

1. This routine cannot be called unless the user's CP directory entry has an XCONFIG ADRSPACE statement with the SHARE parameter.
2. The *user*, *user_list*, and *user_count* parameters must all be supplied. The value specified for option set 2 determines which parameter is used by the DMSSPCP service A 5 (USERID) or 6 (VCIT) causes the *user* parameter to be used and the *user_list* parameter to be ignored. A 7 (USERID LIST) causes the *user_list* to be used and the *user* parameter to be ignored. If 7 (USERID LIST) option is specified, both the *user_list* and *user_count* parameters must contain valid values.
3. When the primary address space of your virtual machine is in a shareable state (through the use of the DMSSPCP routine), subsequent segment loads by DIAGNOSE code X'64' or the CMS SEGMENT LOAD command or macro will be rejected.
4. The use of the READ and WRITE mode specifications can influence the permitted user's use of READ and WRITE mode in the Establish Addressability (DMSSPLA) CSL routine. The DMSSPLA mode specification cannot be more permissive than the specification used in DMSSPCP. For example, READ in DMSSPCP will not allow for WRITE in DMSSPLA; on the other hand, WRITE in DMSSPCP will allow either READ or WRITE to be specified in DMSSPLA.
5. For detailed information on the permit function, see the ADRSPACE PERMIT macro description in the *z/VM: CP Programming Services*.

Return Codes and Reason Codes

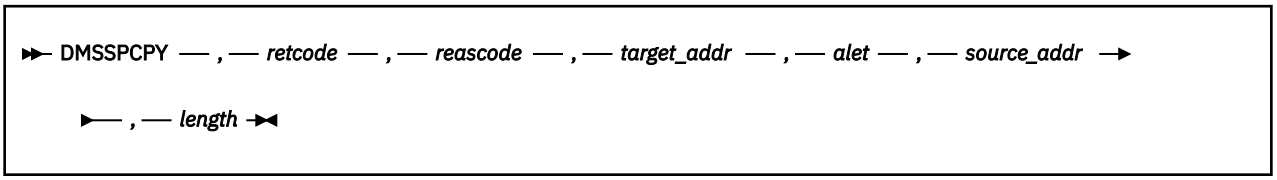
For lists of the possible return codes from DMSSPCP, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPCP. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Severity	Reason Code	Description
WARNING	85024	All virtual machines identified by the *user* or *user_list* parameter have already been authorized to access the address space. There was no change in the virtual machines' authorization.
WARNING	85107	The virtual machine list identified by *user_list* has been processed, but there was at least one user ID that was already authorized or that was not currently logged on.
ERROR	85004	The specified address space identification token (ASIT) does not identify a data space that was created by your virtual machine. This could occur if a data space is not created, or if it is created using DMSSPCC, but then is deleted using a CP macro before calling DMSSPCD.
ERROR	85028	All of the virtual machines identified by the *user* or *user_list* parameter are not either currently logged on or disconnected. No authorization was allowed.
ERROR	85032	The issuer's XCONFIG ADDRSPACE directory statement did not have the SHARE option specified. See the *z/VM: CP Planning and Administration* for more information on directory entries.
ERROR	85100	This support is not provided by the processor on which VM is running.

Severity	Reason Code	Description
ERROR	85102	Incorrect parameter or options specified. The error could be: • An incorrect value • Two options having a duplicate value • One option's value conflicting with another's.
ERROR	85103	The specified address space does not have the CMS SHARE attribute specified to allow access from another virtual machine. No authorization was allowed.
ERROR	85104	Illegal request for a address space service. For example, DMSSPCC was not called to create the data space specified on this permit request.
ERROR	85106	Incorrect value specified for the *user_count* parameter. No user IDs were processed.
ERROR	96100	Insufficient CMS virtual storage was available for this request.

DMSSPCPY - Copy from Address Space



Call Format

The format for calling a CSL routine is language dependent. DMSSPCPY is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCPY
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCPY routine if your virtual machine is an XA virtual machine and you want to copy information from a data space or another user's primary address space into your virtual machine's storage.

If called from an XC virtual machine, DMSSPCPY fails with return code 8 and reason code 85100.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCPY.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCPY.

target_addr
(input, INT, 4) is a variable used for specifying the starting storage address in your virtual machine of the area to which the data from the address space will be copied.

The area of your virtual machine's primary address space into which data from an address space is to be copied either must be a part of a previously loaded program or segment, or must have been previously obtained using the CMSSTOR or GETMAIN macro.

alet
(input, INT, 4) is a variable used for passing the access list entry token (ALET) used to identify the address space to be copied from. See Usage Note "1" on page 496 for the sequence of calls required for obtaining the ALET.

source_addr
(input, INT, 4) is a variable used for specifying the starting storage address in the address space from which the data will be copied.

length
(input, INT, 4) is a variable used for specifying the number of bytes to be copied from the address space to your virtual machine's primary address space.

Specifying a value in the *length* parameter that is greater than the size of the area specified by the *target_addr* parameter may cause unpredictable results due to overlaying CMS storage management control data or other application-related data.

Usage Notes

1. The call to DMSSPCPY must be preceded by the following CSL routine calls:
   a. Permit Address Space Access (DMSSPCP) must be called by the owner of the address space to permit access.
   b. Query Address Space (DMSSPCQ) must be called by your program to obtain the access list identification token (ASIT) of the address space to be accessed, unless the ASIT was previously passed to your program.
   c. Establish Address Space Addressability (DMSSPLA) must be called by your program to add the address space access list entry (ALE) to your virtual machine's access list and to obtain the access list entry token (ALET) for passing in the *alet* parameter of DMSSPCPY.
2. The use of this routine may result in program exceptions that will be intercepted by CMS. The reason code value will contain the program exception interrupt code. For example, Code X'0028' is returned as reason code 85040.
3. For detailed information on the copy function, see the DIAGNOSE Code X'248' description in *z/VM: CP Programming Services*.

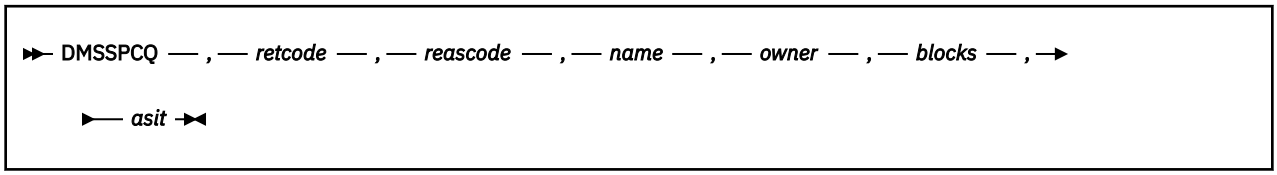
Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCPY, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPCPY. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85008	Error in the parameter specification.
ERROR	85004	Protection exception (Code X'0004') • A part of the target area is in CP-page-protected storage. • The PSW key is not valid for accessing either the source or target storage.
ERROR	85005	Addressing exception (Code X'0005') • The source area extends into storage locations not available in the address space. • The target area extends into storage locations not available in your primary address space.
ERROR	85040	ALET specification exception (Code X'0028') • The *alet* is invalid.
ERROR	85041	ALEN translation exception (Code X'0029') • The *alet* designates an access list entry in an unused state.
ERROR	85100	This support is not available or this service was called from an XC virtual machine.
ERROR	85108	An ESTAE-related problem occurred that could jeopardize the ability to report an interrupt code in the event of a program check.
ERROR	85310	Addressing capability exception (Code X'0136') • The *alet* designates an ALE that is in revoked state.

DMSSPCQ - Query Address Space



Call Format

The format for calling a CSL routine is language dependent. DMSSPCQ is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCQ
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPCQ" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCQ routine to query information about an address space that your virtual machine owns or is authorized to access.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCQ.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCQ.

name
(input, CHAR, 24) is a variable used for specifying the name of the address space for which information is to be extracted by this call. The name is a string of 1 to 24 characters. Only uppercase letters (A through Z), numbers (0 through 9), and certain special characters (# $ @ _ -) are allowed. If the name contains incorrect characters, the query is not performed. The name must be padded on the right with blanks if it is not 24 characters long.

owner
(input, CHAR, 8) is a variable used for specifying the user ID of the virtual machine that owns the address space for which information is being requested. Specifying an asterisk (*) indicates that the virtual machine from which this routine is being called owns the address space for which information is being requested. If the value specified for this parameter is less than 8 characters long, you must pad it on the right with blanks.

blocks
(output, INT, 4) is a variable used for returning the number of 4096-byte blocks in the address space queried.

asit
(output, CHAR, 8) is a variable used for returning the address space identification token (ASIT), the unique system-wide identifier for the address space being queried.

Usage Notes

1. Your application can call DMSSPCQ to obtain the address space identification token (ASIT) of its own primary address space for the purpose of permitting other users to access it, or for releasing pages within the primary address space. Specify BASE for the *name* parameter to obtain the ASIT of your own virtual machine's primary address space.
2. If you specify a user ID (to obtain information about another user's address space), the address space owner must have previously specified your user ID on a call to the Permit (DMSSPCP) CSL routine.
3. For detailed information on the query function, see the ADRSPACE QUERY macro description in *z/VM: CP Programming Services*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCQ, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by DMSSPCQ. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85004	You do not own or have access to the address. No information was returned.
ERROR	85016	The specified address space name contains incorrect characters. No information was returned.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPCR - Restore Address Space Access

Call Format

The format for calling a CSL routine is language dependent. DMSSPCR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCR
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPCR" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCR routine to restore access for previously permitted users after the DMSSPCI (Isolate Address Space) routine has been called. A successful call to DMSSPCI must precede an attempt to restore access.

A single call to DMSSPCR allows virtual machines that were permitted to access the address space before the isolation to re-establish addressability, except that logged-off virtual machines are removed from the permitted list during the restoration operation.

If DMSSPCI was called with NOPURGE specified, DMSSPCR give access back to logged-on virtual machines that were previously given access to the address space by a call to the Permit Address Space Access (DMSSPCP) routine.

If PURGE was specified on the prior call to the DMSSPCI routine, the DMSSPCR call will fail. The users must be specified on the DMSSPCP routine again before they can re-establish addressability to the address space.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCR.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCR.

asit
(input, CHAR, 8) is a variable used for specifying the address space identification token (ASIT), the unique system-wide identifier for the address space for which access is to be restored.

Usage Notes

1. Virtual machines must establish addressability to the address space after access is restored, even if they had established addressability prior to the call to the DMSSPCI routine. See the *z/VM: CMS Application Development Guide* for more information.
2. For detailed information on the restore function, see the ADRSPACE PERMIT macro description in *z/VM: CP Programming Services*.

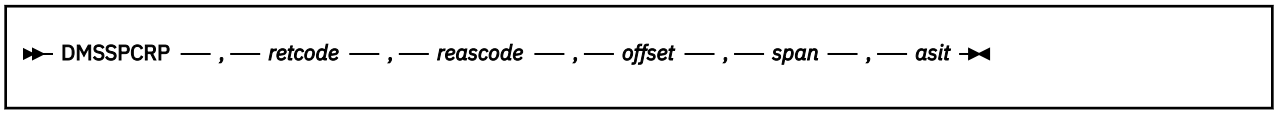
Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCR, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPCR. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

Severity	Reason Code	Description
WARNING	85024	All virtual machines that were previously permitted and represented by CMS structures have already been authorized to access the address space. There was no change in the virtual machines' authorization.
WARNING	85105	A call sequence warning occurred. This call was not preceded by a call to DMSSPCI. Any users that currently had access still have it; any users that were logged off have been removed from the CMS-maintained list of users with access to the address space.
WARNING	85107	The previously permitted virtual machines now have their address space access restored, but there was at least one invalid (for example, logged off) entry detected in the CMS-maintained permit list. Any user found to be logged off is removed from the permit list.
ERROR	85004	The address space identification token specified in the *asit* parameter does not identify an address space owned by the issuing virtual machine. No authorization was allowed.
ERROR	85028	All of the virtual machines that were previously permitted and represented by CMS structures are either not currently logged on or are disconnected. No authorization was allowed.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	85104	Illegal request for a CMS address space service. For example, PURGE was specified on the preceding DMSSPCI routine and therefore, there are no users with permission to access the address space. They must have their access restored with the DMSSPCP routine. This reason code can also mean that DMSSPCC was not called to create the address space designated on this call.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPCRP - Release Address Space Pages



Call Format

The format for calling a CSL routine is language dependent. DMSSPCRP is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPCRP
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPCRP routine to release pages of storage in address space storage that you own. Only the owner of the address space can call this routine.

When this function is used to release an unmapped page, the contents of the page are changed to binary zeros.

When this routine releases a mapped page, the contents of the page on a subsequent reference will reflect the contents of the associated DASD block to which the page was mapped. A page is considered mapped when the DEFINE function of the CP MAPMDISK macro has been used to associate the page with a DASD block on a minidisk.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPCRP.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPCRP.

offset
(input, INT, 4) is a variable used for specifying the offset (in number of pages) of the first page of storage that is to be released. A value of 0 specifies the first storage block of 4096 bytes, or bytes 0-4095 of the address space. A value of 1 specifies that the release will start at the second block, bytes 4096 to 8191, and so on.

span
(input, INT, 4) is a variable used for specifying the number of 4096-byte blocks in the address space to be released.

asit
(input, CHAR, 8) is a variable used for specifying the address space identification token (ASIT), the unique system-wide identifier for the address space to be released.

Usage Notes

1. Addressability to the address space must have been established for WRITE (and be in effect) before calling this routine.
2. To release pages in the caller's primary address space, the ASIT value must be all zeros to identify the primary address space.

3. The use of this routine may result in program exceptions that will be intercepted by CMS. The reason code value will contain the program exception interrupt code. For example, Code X'0028' is returned as reason code 85040.
4. For detailed information on the release pages function, see the DIAGNOSE Code X'10' description in *z/VM: CP Programming Services*.

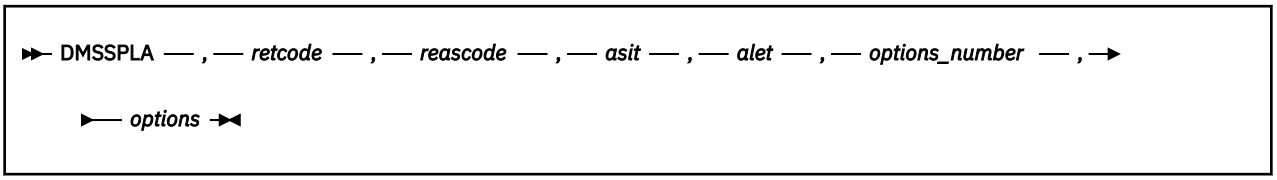
Return Codes and Reason Codes

For lists of the possible return codes from DMSSPCRP, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by DMSSPCRP. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85005	Addressing exception (Code X'0005') • An attempt to release a page outside its defined storage.
ERROR	85006	Specification exception (Code X'0006') • An attempt was made to release page 0 of the primary address space.
ERROR	85040	ALET specification exception (Code X'0028') • The *alet* is incorrect.
ERROR	85041	ALEN translation exception (Code X'0029') • The *alet* designates an ALE (access list entry) in an unused state.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	85104	Illegal request for a CMS address space service. For example, the requesting virtual machine does not own the address space in which pages are to be released.
ERROR	85105	A call sequence error occurred. Addressability has not been established to the address space. Either DMSSPLA was not called prior to this call, or DMSSPLR was called to remove the addressability.
ERROR	85108	An ESTAE-related problem occurred that could jeopardize the ability to report an interrupt code in the event of a program check.
ERROR	85310	Addressing capability exception (Code X'0136') • The *alet* designates an ALE (access list entry) in the revoked state.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPLA - Establish Address Space Addressability



Call Format

The format for calling a CSL routine is language dependent. DMSSPLA is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPLA
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPLA" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPLA routine to establish addressability to an address space by adding a valid access list entry (ALE) to your virtual machine's access list.

This allows your virtual machine to address an address space owned by your virtual machine or by another virtual machine to which you have been permitted access by the DMSSPCP (Permit Address Space Access) routine, called from the address space owner's virtual machine.

For information on using address space and access list services provided in CMS, see *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPLA.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPLA.

asit
(input, CHAR, 8) is a variable used to specify the address space identification token (ASIT), the unique system-wide identifier for the address space to which addressability is being requested.

alet
(output, INT, 4) is used for returning the access list entry token (ALET) that will allow your virtual machine to reference the address space identified by the address space identification token specified in the *asit* parameter.

options_number
(input, INT, 4) is a variable used to specify the number of entries in the array of option values.

options
(input, INT, 4) is the first of an array of variables that contain option values. (In REXX, this is the stem of a compound variable, *without the period*.)

Option values specify the attributes of the address space's access list entry (ALE) in your virtual machine's access list. There are two sets of option values. Only one option value can be specified for each set. If no option values are specified, a zero should be placed in the *options_number* parameter, so the options parameter will be ignored and the default keywords will be used. The valid option sets for this routine are:

Set 1
defines whether the address space can be modified by another program or user. Valid values are:

0 (READ)
specifies that the permitted program or user can only read from the address space identified by the *asit* parameter. This is the default value for this option set.

1 (WRITE)
specifies that the permitted program or user can write to the address space identified by the *asit* parameter.

Specifying the WRITE option for an address space to which you have been permitted read-only access results in a warning return code.

Set 2
defines whether there will be notification of page faults occurring in the address space specified by the ALET. Valid values are:

5 (SYNC)
specifies that there is to be no notification of page faults given to the caller of this routine. This is the default value for this option set.

6 (ASYNC)
specifies that the caller of this routine is to be notified of page faults occurring in the address space specified by the ALET. This notification will be presented asynchronously through a page-fault notification interrupt. For details, see the PFAULT macro description in *z/VM: CP Programming Services*.

Only applications executing in an XC virtual machine can specify the ASYNC option when establishing addressability to an address space to which it has been granted access. If an application running in an XA virtual machine specifies the ASYNC option, the request is ignored.

Usage Notes

1. The DMSSPLA mode specification cannot be more permissive than the specification used in DMSSPCP by the owner of the address space. For example, READ in DMSSPCP will not allow for WRITE in DMSSPLA; on the other hand, WRITE in DMSSPCP will allow either READ or WRITE to be specified in DMSSPLA.
2. If you specify the ASYNC option, your program must also establish a HNDEXT interrupt handler exit and specify a token address on the CP PFAULT macro.
3. After your application has specified the ASYNC option, it can cancel unwanted unresolved page fault notifications by calling the CP PFAULT macro with the CANCEL function. Any pending page fault completion notifications will be purged.
4. See *z/VM: CMS Application Development Guide* and *z/VM: CP Programming Services* for more information on how to use the ASYNC option.
5. You need not establish addressability to your own primary address space with this routine, even though you used the Create Data Space (DMSSPCC) routine to establish the structures that permit other users to access it.
6. For detailed information on the establish addressability function, see the ALSERV ADD macro description in *z/VM: CP Programming Services*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSSPLA, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPLA. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85004	The address space identification token specified for the *asit* parameter does not identify a valid address space (one owned by the caller or one to which the caller has been permitted access). The ALET is not returned and addressability is not established to the address space.
ERROR	85008	There are no available entries in the access list which can be used for the current request. The ALET is not returned and access is not allowed to the address space.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	85102	Incorrect parameter or options specified. The error could be: • An incorrect value • Two options having a duplicate value • One option's value conflicting with another's.
ERROR	96100	Insufficient CMS virtual storage was available to perform the service.

DMSSPLR - Remove Address Space Addressability

Call Format

The format for calling a CSL routine is language dependent. DMSSPLR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSPLR
(input, CHAR, 8) can be passed as a literal or in a variable. "DMSSPLR" must be padded with blanks to eight characters.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSSPLR routine to remove addressability to an address space by changing the state of a valid or revoked access list entry (ALE) on your virtual machine's access list to unused.

Upon completion of this routine, the address space cannot be addressed with the ALE.

For information on using address space and access list services provided in CMS, see the *z/VM: CMS Application Development Guide*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSPLR.

reascode
(output, INT, 4) is a variable for the reason code from DMSSPLR.

alet
(input, INT, 4) is a variable used for specifying the ALE to be changed to an unused state. This value was returned by the Establish Address Space Addressability (DMSSPLA) CSL routine when addressability to the address space was established.

An ALET of X'00000000' designates the primary address space of your virtual machine and does not designate any ALE. An attempt to use DMSSPLR with this value ALET results in a return code of 8 and a reason code of 85012.

Usage Notes

1. The ALE that was marked as unused by this service will be available for subsequent calls to DMSSPLA.
2. For detailed information on the remove addressability function, see the ALSERV REMOVE macro description in *z/VM: CP Programming Services*.

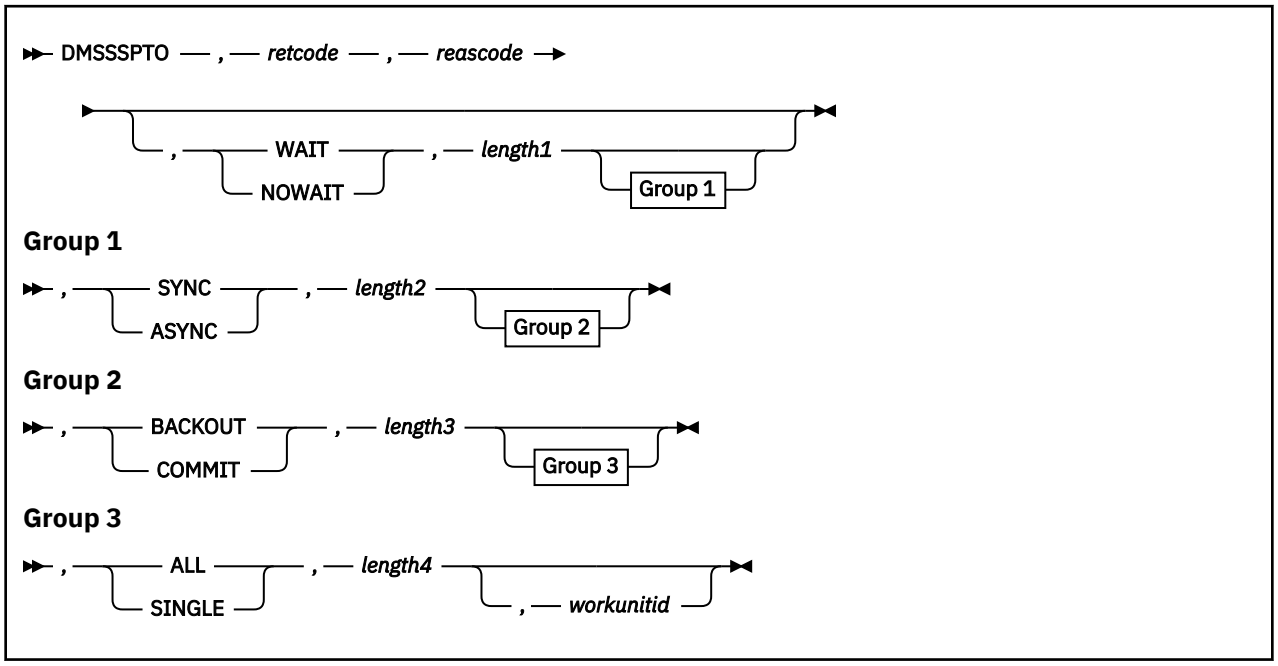
Return Codes and Reason Codes

For lists of the possible return codes from DMSSPLR, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by DMSSPLR. ERROR indicates that the request failed. Errors cause return code 8 or 12.

Severity	Reason Code	Description
ERROR	85004	The ALE designated by the access list entry token (ALET) was already in an unused state. The state of the ALE is unchanged.
ERROR	85012	The ALET specified in the *alet* parameter is not valid. The state of the ALET is unchanged.
ERROR	85100	This support is not provided by the processor on which VM is running.
ERROR	96100	Insufficient CMS storage was available to perform the service.

DMSSSPTO - Set Synchronization Point Options



Context

Coordinated Resource Recovery

Call Format

The format for calling a CSL routine is language dependent. DMSSSPTO is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSSSPTO
(input, CHAR, 8) can be passed as a literal or in a variable.

Purpose

Use the DMSSSPTO routine to set various synchronization point options for Coordinated Resource Recovery (CRR).

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSSSPTO.

reascode
(output, INT, 4) is a variable for the reason code from DMSSSPTO.

WAIT
(input, CHAR, 4) indicates that if resynchronization is required, control will not return to your application until resynchronization completes. Omitting this parameter is the same as specifying WAIT.

NOWAIT
(input, CHAR, 6) indicates that if resynchronization is required, resynchronization will be attempted before returning control to your application. If it is not possible to complete resynchronization on the

first attempt, control is returned to your application. Omitting this parameter is the same as specifying WAIT.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character variable (WAIT or NOWAIT). Specifying 0 is equivalent to selecting WAIT.

SYNC
(input, CHAR, 4) indicates that CMS will communicate synchronously with a protected resource when it is more efficient to do so.

ASYNC
(input, CHAR, 5) indicates that CMS will communicate asynchronously with protected resources, issuing a DMSCWAIT (CRR Wait) for any requests.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character variable (SYNC or ASYNC). Specifying 0 is equivalent to selecting SYNC.

BACKOUT
(input, CHAR, 7) indicates that CMS should roll back the changes to the specified work units if a protocol violation is received from the synchronization point initiator while in prepared state. Omitting this parameter is the same as specifying BACKOUT.

COMMIT
(input, CHAR, 6) indicates that CMS should commit the changes to the specified work units if a protocol violation is received from the synchronization point initiator while in prepared state. Omitting this parameter is the same as specifying BACKOUT.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character variable (BACKOUT or COMMIT). Specifying 0 is equivalent to selecting BACKOUT.

ALL
(input, CHAR, 3) indicates that the selections specified for the WAIT/NOWAIT, SYNC/ASYNC, and BACKOUT/COMMIT options are to be set as the default values for all work units that are not specified individually, both present and future. Omitting this parameter is the same as specifying ALL.

SINGLE
(input, CHAR, 6) indicates that the selections specified for the WAIT|NOWAIT, SYNC|ASYNC, and BACKOUT|COMMIT options are to be set for the work unit specified in the *workunitid* parameter. Omitting this parameter is the same as specifying ALL.

length4
(input, INT, 4) is a variable for specifying the length of the preceding character variable (ALL or SINGLE). Specifying 0 is equivalent to selecting ALL.

workunitid
(input, INT, 4) is a variable you can use to specify the work unit to be associated with the synchronization point options. If the *workunitid* parameter is omitted and SINGLE is specified, the options are set for the current default work unit.

Usage Notes

1. If you specify ALL, the *workunitid* parameter is ignored.
2. If you specify SINGLE, then you must have previously obtained the specified work unit ID with DMSGETWU (Get Work Unit ID) or you must omit or specify 0 for the *workunitid* to indicate that the current default work unit is to be used.
3. If DMSSSPTO is never issued from your virtual machine, then the options used will be WAIT, SYNC, BACKOUT and ALL.
4. If you call DMSSSPTO several times, then:
   - If the first invocation is for SINGLE, later invocations for ALL will not affect the options set for the work unit ID specified on the first call for SINGLE.

- If the first invocation is for ALL, later invocations for SINGLE will set the new options selected for the specified work unit ID.

5. If you specify ALL or nothing, the selected options remain in effect until the end of the CMS session (or until a subsequent DMSSSPTO for ALL is specified).
6. If you specify SINGLE, the selected options remain in effect until the work unit is returned to CMS. Work units are returned to CMS and the resources associated with them are freed at the following times:
   - At end-of-command, all work units associated with the command are returned.
   - When a CMS ABEND occurs, all work units are returned.
   - At end of CMS SUBSET, all subset mode work units are returned.
   - When the DMSRETWU (Return Work Unit ID) CSL routine is invoked, the specified work unit is returned.
   - When the DMSPURWU (Purge Work Unit IDs) CSL routine is invoked, all work units are returned.
7. To avoid being put into a wait state by CMS when communicating with protected resources, a multiuser server application using the ASYNC parameter should provide its own CRR Wait routine. For information about replacing DMSCWAIT, see *Application Development Guide*.
8. Issuing DMSSSPTO from the system profile effectively changes the system defaults.

Return Codes and Reason Codes

For lists of the possible return codes from DMSSSPTO, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSSPTO. ERROR indicates that the request failed. Errors cause return code 8.

Severity	Reason Code	Description
ERROR	78020	Invalid synchronous option specified; must be SYNC or ASYNC.
ERROR	78021	Invalid commit decision option specified; must be BACKOUT or COMMIT.
ERROR	78022	Invalid work unit option specified; must be ALL or SINGLE.
ERROR	90477	Invalid wait option specified; must be WAIT or NOWAIT.
ERROR	90540	Invalid work unit ID.
ERROR	96100	Insufficient virtual storage.

DMSSTKC / StackBufferCreate - Add a Buffer to the Program Stack

See “StackBufferCreate / DMSSTKC - Add a Buffer to the Program Stack” on page 558.

DMSSTKD / StackBufferDelete - Drop Buffers from the Program Stack

See “StackBufferDelete / DMSSTKD - Drop Buffers from the Program Stack” on page 560.

DMSSTKQ / StackQuery - Query the Program Stack

See “StackQuery / DMSSTKQ - Query the Program Stack” on page 562.

DMSSTKR / StackRead - Read from the Program Stack

See “StackRead / DMSSTKR - Read from the Program Stack” on page 564.

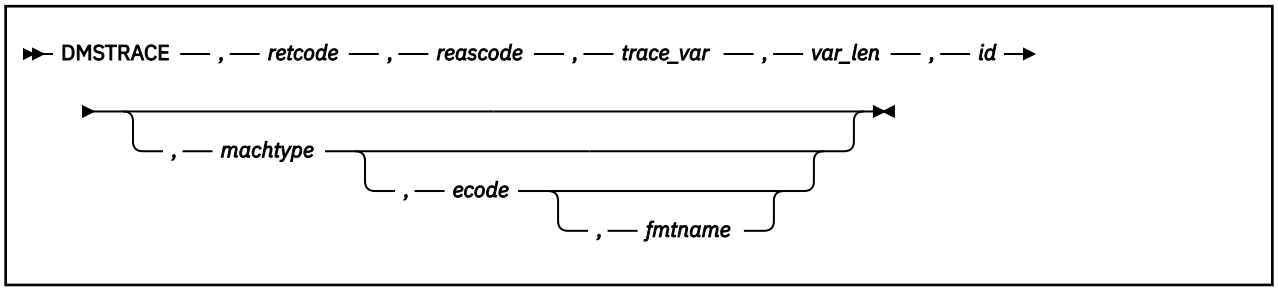
DMSSTKW / StackWrite- Write to the Program Stack

See “StackWrite / DMSSTKW - Write to the Program Stack” on page 567.

DMSTCD / VMTCPDT - Parse TCPIP DATA File

See “VMTCPDT / DMSTCD - Parse TCPIP DATA File” on page 569.

DMSTRACE - Trace



Call Format

The format for calling a CSL routine is language dependent. DMSTRACE is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSTRACE
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSTRACE routine to record trace data using the external trace facility available on CP. You can use the external trace records to help find program problems or improve performance. See Usage Note "1" on page 518 for information on enabling tracing before using this routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSTRACE.

reascode
(output, INT, 4) is a variable for the reason code from DMSTRACE.

trace_var
(input, CHAR, 4) is a variable for specifying the data to be recorded when this routine is encountered during execution.

var_len
(input, INT, 4) is a variable for specifying the length (1 - 2048) of the trace data (contents of the variable specified in *trace_var*) to be recorded.

id
(input, INT, 4) is a variable for specifying the code (0 - 65535) supplied by the application. The trace routine records this code with the trace data, making it easier to find a trace entry.

machtype
(input, CHAR, 1) is a 1-byte code (0 - FF) used to identify individual virtual machine type records. This parameter is supported by the TRSOURCE command in BLOCK mode. Established codes are:

X'01'
TSAF virtual machine entries

X'02'
File pool server machine

X'03'
PVM

X'04'
CICS/VM

X'05' - X'FD'
Reserved for IBM use

X'FE'
Field engineering entries

X'FF'
User installation entries

To use this machine type, you must provide a format routine named USERFMT and assemble it as USERFMT TEXT. It must conform to the same interface guidelines as the *fmtname* parameter.

ecode
(input, CHAR, 8) is a variable for specifying a code (assigned by you) to further identify information. This parameter is supported by the TRSOURCE command in BLOCK mode.

fmtname
(input, CHAR, 8) is a variable for specifying the format exit routine identifier. If no name is provided, records are formatted by the default formatting routine for the machine type. If the machine type is also not specified, records are formatted in the standard hexadecimal dump format. This option is supported by the TRSOURCE command in BLOCK mode.

Be sure that a format routine with the specified name is available to format the trace records. The format routine can have either long or short format. The address of the parameter list shown in usage note "3" on page 518 is passed to the format routine in Register 1. For more information, see the TRACERED formatting exit interface in *z/VM: Dump Viewing Facility*.

Usage Notes

1. To use DMSTRACE, you must first enter the TRSOURCE command to enable tracing at the CP level. Then you must enter the CMS ETRACE command to enable CMS tracing.

   TRSOURCE is a class A, C, or E command. If you do not have class authority to use the one you need, contact your system administrator to have the command entered for you.
2. Ending the tracing also requires a certain sequence of disabling. First enter the CMS ETRACE command to end the tracing in the virtual machine and then enter the TRSOURCE command to disable the tracing at the CP level.
3. The trace formatter parameter list is:

   Offset
   Data

   X'0'
   Address of the trace record

   X'4'
   Address of the trace record data portion

   X'8'
   Length of the trace record

   X'C'
   Length of the trace record data portion

   X'10'
   Reserved, set to zero

   X'14'
   Address of the output buffer, into which the exit will build a line of formatted output. It is initially blank.

   X'18'
   Reserved, set to zero

X'1C'
Address of the output routine (in the TRACERED program).

X'20'
Communications Flags

1... ....
Output should be formatted in 80 byte records for display. This is set when TRACERED is invoked with the CMS operand, or when the output length is overridden by the LRECL 80 option.

.1.. ....
Output should be formatted in 132 byte records for printing. This is set when TRACERED is invoked with the PRINT operand, or when the output length is overridden by the LRECL 132 option.

..1. ....
Reserved, set to zero

...1 ....
Long output format wanted. The FORMAT option was specified (or defaulted) on the command. This bit is zero if the HEX option was specified.

.... 1111
Reserved, set to zero.

4. The following messages can also be encountered when using DMSTRACE, although this routine does not issue them itself:

DMSABE1329I
Trace records were lost [RC=0]

DMSDIE1329I
Trace records were lost [RC=0]

DMSITP1333E
TRSOURCE is disabled. Record not added to buffer [RC=0]

DMSITP1333E
TRSOURCE is in EVENT mode. Record not added to buffer [RC=0]

DMSITP1333E
ETRACE is disabled. Record not added to buffer [RC=0]

DMSITP1333S
I/O or severe error. Buffer not written [RC=0]

DMSITP1334E
Buffer not initialized. Reissue ETRACE [RC=0]

DMSITP1335E
Incorrect Monitor Call Class 10 Code [RC=0]

Return Codes and Reason Codes

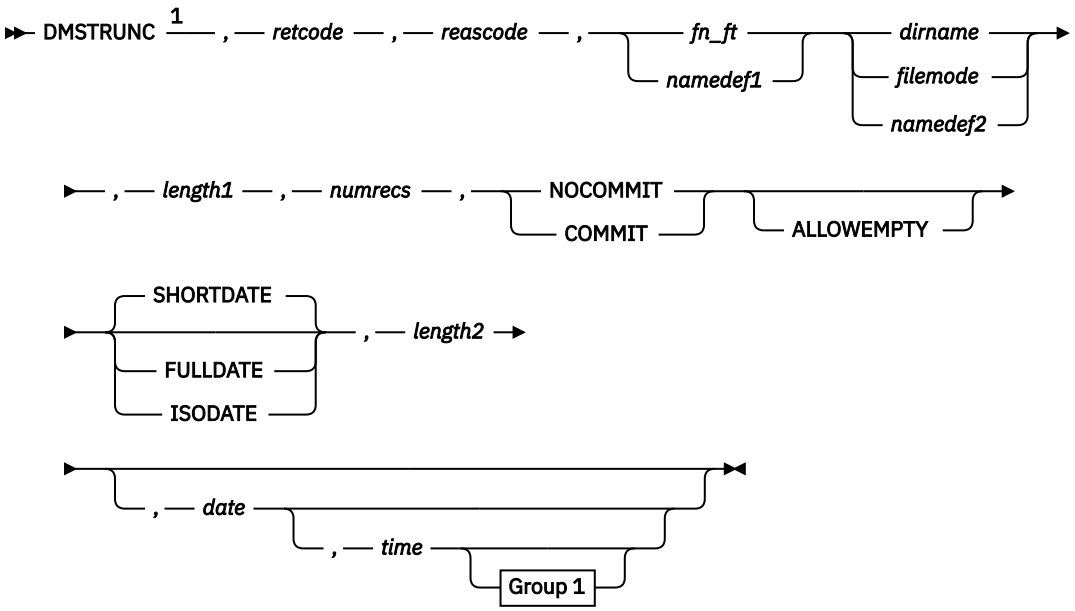
For lists of the possible return codes from DMSTRACE, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by DMSTRACE. ERROR means the request failed. Errors cause return code 8.

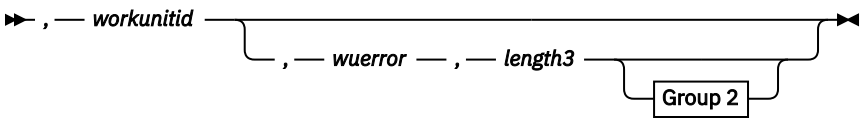
Severity	Reason Code	Description
ERROR	99001	Invalid value specified for the *datalen* parameter. It was not in the range of 0 to 2048.
ERROR	99002	Invalid value specified for the *id* parameter. It was not in the range of 0 to 65535.

Severity	Reason Code	Description
ERROR	99003	Invalid value specified for the *machtype* parameter. It was not in the range of 0 to 255.
ERROR	99004	The DMSTRACE facility (Monitor Call Class 10) is not enabled.

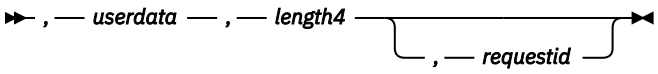
DMSTRUNC - Truncate



Group 1



Group 2



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Context

File System Management: SFS and minidisk

Call Format

The format for invoking a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSTRUNC, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSTRUNC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSTRUNC routine to delete records from the end of an existing minidisk or SFS file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSTRUNC.

reascode
(output, INT, 4) is a variable for the reason code from DMSTRUNC.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of the file being truncated.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying the SFS directory containing the file being truncated.

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed SFS directory or minidisk. If this file mode letter is also a one-character temporary name (*namedef2*), the temporary name is used.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously created for a *dirname* or *filemode*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*; plus *dirname*, *filemode*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

numrecs
(input, INT, 4) specifies the number of records you want the resultant file to be. This value must be less than or equal to the number of records already in the file. If a value of zero is specified, the file must reside on SFS and ALLOWEMPTY must be specified. In this case, the existing file is replaced with an empty file. An error will result if zero is specified and ALLOWEMPTY is not specified.

COMMIT
(input, CHAR, 6) means to keep all changes associated with the work unit (including changes made to other files), from either the start of the work unit or the last commit. See the COMMIT option description under "Common Parameters" on page 15 for more information.

NOCOMMIT
(input, CHAR, 8) means do not commit changes.

ALLOWEMPTY
(input, CHAR, 10) permits a file with no records to be created by this DMSTRUNC request if the number of records parameter (*numrecs*) is specified as zero.

This parameter is not applicable for a minidisk file. Empty files are not allowed on minidisks. A warning will result if ALLOWEMPTY is specified for a minidisk file.

SHORTDATE
(input, CHAR, 9) indicates the format of the *date* parameter is *yy/mm/dd*, where yy is the 2-digit year, *mm* is the month, and *dd* is the day of the month. SHORTDATE is the default.

FULLDATE
(input, CHAR, 8) indicates the format of the *date* parameter is *yyyy/mm/dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

ISODATE
(input, CHAR, 7) indicates the format of the *date* parameter is *yyyy-mm-dd*, where *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

length2
(input, INT, 4) is a variable containing the length of the preceding compound character parameter (COMMIT or NOCOMMIT; ALLOWEMPTY, if specified; or SHORTDATE, FULLDATE, or ISODATE, if specified). See "Compound Variables" on page 15 for coding details.

date
(input, CHAR, 8 or 10) is the *date* attribute that will be saved with the file. The *date* attribute is the date that the file was last updated. It is a character variable with a length of 8 or 10 in the form *yy/mm/dd*, *yyyy/mm/dd*, or *yyyy-mm-dd*, where *yy* is the 2-digit year, *yyyy* is the 4-digit year, *mm* is the month, and *dd* is the day of the month.

You must specify either the FULLDATE or the ISODATE parameter to specify 4-digit years.

You can have the system determine the date by omitting the variable or by specifying 8 blanks for SHORTDATE, or 10 blanks for FULLDATE or ISODATE. The date is in local time.

Use a slash (/) as the separator character to separate the year, month, and day, unless you have specified the ISODATE parameter, which requires a dash (–) separator. For example, the SHORTDATE format of 3 May 1996 is specified as:

```
96/05/03
```

the FULLDATE format is specified as:

```
1996/05/03
```

and the ISODATE format is specified as:

```
1996-05-03
```

time
(input, CHAR, 8) is the *time* attribute that will be saved with the file. The *time* attribute is the time at which the file was last modified. The contents of this variable must be specified in the form *hh:mm:ss*, where *hh* is the hour of the day in 24-hour notation, *mm* is the minutes, and *ss* is the seconds. To use the system-determined time, either omit this parameter or specify eight blanks.

workunitid
(input, INT, 4) is a variable for specifying the work unit to be used for this operation. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length3*) is a variable used to specify an area for returning extended error information. If it is specified, it must be followed by a length field. See *wuerror* under “Common Parameters” on page 15 for more information.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and will result in an error return code.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data (see the Usage Notes).

length4
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*userdata*). The value of *length4* must not be greater than 80. Specifying 0 has the effect of omitting the *userdata* parameter.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid* and is passed on a later Check (DMSCHECK) request.

The processing of a given DMSTRUNC request may not require communication with the server. In this case, the operation is performed synchronously regardless of the value specified in *requestid* and the value of the *requestid* parameter is not changed.

Usage Notes

1. If the file is in SFS, the caller may not have the file open for NEW, WRITE, REPLACE, or CREATEMIG at the time the truncate operation is initiated. An SFS file may be open for READ. A minidisk file may not be open for any intent when Truncate is initiated.
2. If the file is on a minidisk, the ALLOWEMPTY parameter is not applicable. Empty files can be created only in SFS. If *numrecs* is zero, an error will result.
3. If the file is in SFS, the file pool may not support empty files. In this case, if *numrecs* is zero and ALLOWEMPTY is specified, an error will result indicating the level of the file pool server does not support empty files.
4. If your installation does not use an external security manager (ESM), the *userdata* parameter is not used.

   If your installation does use an ESM, refer to the ESM documentation to determine whether you must specify *userdata*. The ESM documentation should also define the format and meaning of the data. The ESM might, for example, require you to specify a password for the object you are truncating. If the directory or file being truncated is on a minidisk, the *userdata* is not used.
5. If a file or directory is open on a work unit and COMMIT is specified, the file remains open and all changes are committed. Current read and write pointers are not changed. There are, however, some situations that can prevent changes from being committed:
   - A CMS Level 6 file pool server that does not have the Commit Without Close Enhancement is involved in the work unit.
   - SFS files that have been modified using the DMSWRBLK (Write Blocks) routine remain open on the work unit.
   - SFS Catalogs (opened by the DMSOPCAT (Open Catalog) routine) remain open on the work unit.
   - SFS files opened using the DMSOPDBK (Open Data Block) remain open on the work unit.
6. If the COMMIT parameter is specified and the return code is 8, then either:
   - An error occurred during the processing of the Truncate operation, or
   - The operation was completed but the work unit could not be committed. In this case the reason code is 50500. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
7. For an asynchronous request, a return code is given indicating whether the request was accepted for processing or was immediately rejected.
8. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
9. For a minidisk file, when the file is truncated all updates to the file have been made. If COMMIT was specified, the work unit will be committed. If an error occurs attempting to commit the work unit, the operations associated with that work unit will remain uncommitted but the minidisk file will be truncated. In this case, a warning return code will be given along with the reason code that indicates why the work unit was not committed.
10. All minidisk requests are done synchronously.
11. Only one of the three date format parameters (SHORTDATE, FULLDATE, or ISODATE) can be specified. SHORTDATE is the default. The date format chosen applies to the *date* parameter.
12. When *date* is specified with the FULLDATE or ISODATE parameters, the 4-digit year (*yyyy*) range is restricted to the range 1900-2099 (that is, the century portion of *yyyy* must be either 19 or 20).

13. When *date* is specified with the SHORTDATE parameter, the sliding window technique is used to calculate a 4-digit year (*yyyy*) from the 2-digit year that is input. The 4-digit year will then be associated with the file.

    The calculation assumes the 2-digit year is within the 100 year window as calculated by:

    (current_year - 50) = low end of window
    (current_year + 49) = high end of window

    For example, if a 2-digit year of 05 is supplied, and the current year (current_year) is 1997, the window range is between the years 1947 and 2046. In this case, the calculation changes the 2-digit year of 05 to the 4-digit year 2005.
14. If you want to perform arithmetic or conversion operations on the time stamps that are input to this routine, you may find the DateTimeSubtract CSL routine helpful. See *z/VM: CMS Application Multitasking*.

Return Codes and Reason Codes

For lists of the possible return codes from DMSTRUNC, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSTRUNC. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

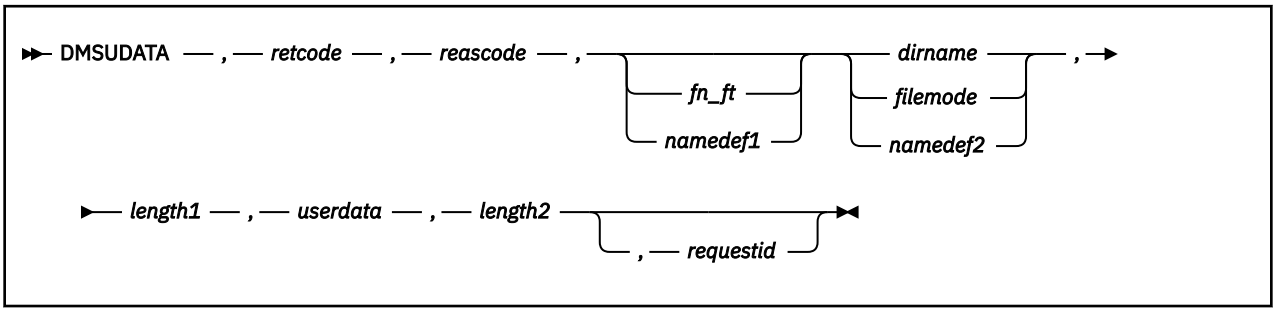
DMSTRUNC can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	90622	ALLOWEMPTY parameter was specified for a minidisk file.
ERROR	31000	You cannot mix recoverable and nonrecoverable work for the same file within a single work unit. The file is nonrecoverable and has been changed without being committed. To truncate the file with DMSTRUNC, first close and commit it.
ERROR	44000	The file does not exist or you are not authorized to write to the file, or the directory does not exist.
ERROR	44200	The file is already open for WRITE, NEW, or REPLACE using DMSOPEN or DMSOPDBK.
ERROR	50500	The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the “Return Codes and Reason Codes” on page 73 for descriptions of these codes.
ERROR	50700	There is no room in the file space to complete this request.
ERROR	63700	Directory control directory is accessed read-only.
ERROR	64300	The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.
ERROR	65200	File is migrated and implicit RECALL is set to OFF. Set RECALL ON or issue a DFSMS RECALL command.
ERROR	65400	Specified operation cannot be performed on an external object or on a BFS file.

Severity	Reason Code	Description
ERROR	65500	Input file is migrated and there is no active SMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)
ERROR	66100	Input file has been migrated and DFSMS/VM recall processing has been disabled.
ERROR	71200	The server encountered an error when it attempted to access a file.
ERROR	90131	Insufficient minidisk space available.
ERROR	90290	Incorrect number of records: Value must be between zero and the number of records currently in the file; or value was zero and ALLOWEMPTY was not specified for an SFS file.
ERROR	90300	Illegal parameter specified. The COMMIT or NOCOMMIT parameter was missing or specified incorrectly, or the ALLOWEMPTY, SHORTDATE, FULLDATE, or ISODATE parameters were specified incorrectly, or extraneous parameters were found.
ERROR	90320	Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.
ERROR	90330	Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, ALLOWEMPTY, SHORTDATE, FULLDATE, and ISODATE.
ERROR	90350	Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least two and not more than three tokens in the string.
ERROR	90410	Incorrect length specified for one of the character variables.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90450	Incorrect characters (* or %) were found in either the file name or file type part of the *fn_ft* parameter.
ERROR	90472	Incorrect request ID specified; it must be 0 or 1.
ERROR	90494	Incorrect date format. The date must be specified in one of the following formats: • *yy/mm/dd* if SHORTDATE is specified • *yyyy/mm/dd* if FULLDATE is specified • *yyyy-mm-dd* if ISODATE is specified
ERROR	90495	Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Severity	Reason Code	Description
ERROR	90496	The date specified is incorrect; it must be a number 0-9.
ERROR	90498	The time specified is in incorrect format; it must be in the format *hh:mm:ss*.
ERROR	90499	The time specified is incorrect; it must be a number 0-9.
ERROR	90500	The specified directory name is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90540	Specified work unit ID is incorrect.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.
ERROR	90603	Attempted to truncate the file and the file mode is read-only.
ERROR	90604	Minidisk file already open using the FS macro interface.
ERROR	90606	I/O error found when processing a minidisk file.
ERROR	90615	The specified disk must be a CMS-formatted minidisk.
ERROR	90680	I/O error accessing OS dataset.
ERROR	90681	OS read password protected dataset.
ERROR	90682	OS dataset organization is not BSAM, QSAM or BPAM.
ERROR	90683	OS dataset more than 16 extents.
ERROR	90684	Attempt to open a file on an OS or DOS formatted minidisk.
ERROR	90685	Received an unexpected return code while opening a minidisk file.

DMSUDATA - Send User Data



Context

File System Management: SFS and BFS

Call Format

The format for calling a CSL routine is language dependent. DMSUDATA is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSUDATA
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSUDATA routine to pass information to an external security manager (ESM).

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSUDATA.

reascode
(output, INT, 4) is a variable for the reason code from DMSUDATA.

fn_ft
(input, CHAR, 3-17) is a variable for specifying the file name and file type of a file for which you are specifying ESM user data.

namedef1
(input, CHAR, 1-16) is a variable for specifying a temporary name previously defined for *fn_ft*.

dirname
(input, CHAR, 1-153) is a variable for specifying a directory name. If *fn_ft* or *namedef1* is also specified, *dirname* identifies the directory that contains the file. In this case, the ESM user data is associated with the file, not the directory. If *fn_ft* and *namedef1* are omitted, however, the ESM user data is associated with the directory (not with a file).

filemode
(input, CHAR, 1) is a variable for specifying the file mode of an accessed directory. If a one character namedef is used in this field, it is treated as a namedef rather than as a file mode letter. If the file mode letter is not an accessed SFS directory, an error return code will result.

namedef2
(input, CHAR, 1-16) is a variable for specifying a temporary name previously defined for *dirname* or *filemode*.

length1
(input, INT, 4) is a variable for specifying the length of the preceding compound character parameter (*fn_ft* or *namedef1*, if specified; plus *dirname*, *filemode*, or *namedef2*). See "Compound Variables" on page 15 for coding details.

userdata
(input, CHAR, 1-80) is a variable for specifying a string of up to 80 characters of user data to be passed to an external security manager (ESM). The ESM defines the format and meaning of the data.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter.

requestid
(input/output, INT, 4) is a variable that identifies a specific asynchronous request. If it is omitted or contains binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

Usage Notes

1. DMSUDATA is an atomic request. When it is completed, the work in the affected file pool is committed without coordination.
2. If you receive an error reason code from DMSUDATA, it is still possible that DMSUDATA was successful. Suppose, for example, the file pool server receives the DMSUDATA request, passes the user data to the external security manager, and then fails. In this case, your program will receive an error code because the server failed. The server did pass the data to the ESM before it failed, however, so it is likely that the ESM processed the data successfully. But, because the server failed, the ESM could not communicate its success to the server and back to your program. Your program, instead, receives an error code that indicates the server is not available.

   Your ESM may anticipate these conditions and may require additional processing of your program should you receive certain reason codes. Refer to the documentation supplied with your ESM.
3. The file or directory specified does not have to exist in the file pool. CMS does check to ensure that the name is valid (contains only valid characters, for example).
4. A return code of 0 or 4 for an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 for a synchronous request indicates that the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSUDATA, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSUDATA. ERROR means the request failed. Errors cause return code 8 or 12.

DMSUDATA can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	44400	The external security manager (ESM) returned an error to the server machine for this routine. Refer to your ESM documentation to determine why the ESM returned an error to the server (perhaps the *userdata* was not correct). This reason code is also returned if the external security manager does not support ESM user data.

Severity	Reason Code	Description
ERROR	44700	There is no active external security manager. (The file pool server is running with the NOESECURITY start-up parameter in effect.)
ERROR	90350	Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least one and not more than three tokens in the string.
ERROR	90410	Incorrect length specified for one of the character parameters.
ERROR	90420	The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.
ERROR	90430	The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.
ERROR	90450	Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	90500	The specified dirname is incorrect.
ERROR	90505	The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.
ERROR	90510	The *namedef* part of the *fileid* or *dirname* parameter is longer than 16 characters.
ERROR	90530	The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.
ERROR	90590	There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.
ERROR	90601	Input file mode letter did not represent an accessed SFS directory.

DMSUNREG - CRR Resource Adapter Unregistration

Context

Coordinated Resource Recovery (CRR) Participation

Call Format

The format for calling a CSL routine is language dependent. DMSUNREG is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSUNREG
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSUNREG routine to remove an instance of registration (resource and resource adapter pair for a specific work unit) from the CRR synchronization point manager's coordination list. The resource adapter calls this routine when there is no longer any work for the resource requiring coordination through sync point processing. Resource adapters must remove a registration as part of their end-of-work-unit processing.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSUNREG.

reascode
(output, INT, 4) is a variable for the reason code from DMSUNREG.

registry_token
(input, CHAR, 8) is a variable that identifies the registration to be deleted. This value must have been previously returned from a successful call to DMSREG, or unpredictable results may occur.

Usage Notes

1. For guidance information on using the DMSUNREG routine in the context of getting a resource manager to participate in CRR, see *z/VM: CMS Application Development Guide*.
2. If a resource is not in work, and the resource adapter temporarily does not need to participate in sync point activities, the resource adapter should not use DMSUNREG to remove a registration, but instead should use the DMSCHREG (Change Registration) routine to modify the function flags so the resource adapter is not called for SPM sync point functions (Precoordination, Coordination, and PostCoordination). When the resource adapter needs to participate again, it can call DMSCHREG to reset the function flags.
3. The resource adapter may remove a registration if it detects that the resource is no longer available, or coordination of updates is no longer required.
4. Removal of a registration takes effect immediately. If this routine is called while processing a sync point, the resource adapter is not redriven or involved in any later sync point processing, such as postcoordination.

5. Removing a registration for a resource does not delete its error data. Error data is preserved until the start of the next sync point or until the work unit ends. See “DMSGETER - CRR Get My Errors” on page 271.

Return Codes and Reason Codes

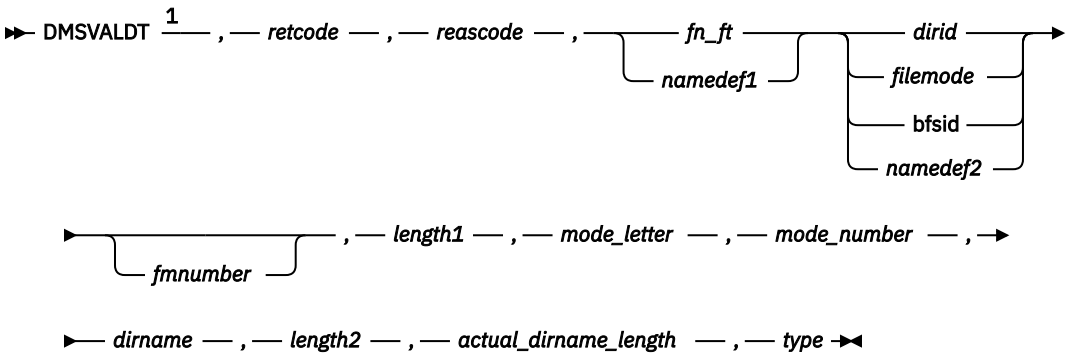
For lists of the possible return codes from DMSUNREG, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSUNREG. ERROR means the request failed. Errors cause return code 8.

DMSUNREG can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	78003	Invalid *registry_token* parameter.

DMSVALDT - Validate



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for invoking a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSVALDT, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSVALDT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSVALDT routine to validate a file ID. DMSVALDT indicates whether the file ID is valid and returns information about it.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSVALDT.

reascode
(output, INT, 4) is a variable for the reason code from DMSVALDT.

fn_ft
(input, CHAR, 3-17) is the file name and file type to be validated. For a BFS file, these are system-generated values.

namedef1
(input,CHAR, 1-16) is a temporary name previously created for a *fn_ft*.

dirid
(input, CHAR, 1-153) is the ID of an SFS directory. The syntax of *dirid* is explained in the *z/VM: CMS Commands and Utilities Reference*.

filemode
(input, CHAR, 1) is a file mode letter of an accessed SFS directory or minidisk. If this file mode letter is also a one-character temporary name, it is treated as a temporary name (*namedef2*) rather than a file mode.

bfsid
(input, CHAR, 1-18) is the name of the byte file system containing the file to be validated.

namedef2
(input, CHAR, 1-6) is a temporary name previously created for a *dirid*, *filemode*, or *bfsid*.

fmnumber
(input, CHAR, 1) is a file mode number. It is a character value between '0' and '6', inclusive.

length1
(input, INT, 4) is the length of the preceding character parameter (*fn_ft* or *namedef1*; plus *dirid*, *filemode*, *bfsid*, or *namedef2*; plus *fmnumber*, if specified).

mode_letter
(output, CHAR, 1) is the file mode letter of the input minidisk or SFS directory:

- When a mode letter is specified as input (*dirid*, *filemode*, or *namedef2*) that letter is returned in *mode_letter*.
- When an SFS directory name is specified as input (*dirid* or *namedef2*), *mode_letter* is the first file mode at which the directory is accessed.
- When a BFS top directory name is specified as input (*bfsid*), when a specified SFS directory is not accessed, or when *namedef2* resolves to an unused file mode, *mode_letter* returns a blank.

mode_number
(output, CHAR, 1) is the file mode number of the file. When *fmnumber* is not specified, *mode_number* is a blank.

dirname
(output, CHAR, 1-153) is a fully qualified directory name or a blank. It is a blank for a minidisk file or when no directory is accessed at the specified mode.

length2
(input, INT, 4) is the length of the preceding character parameter (*dirname*), which must be at least as long as the directory name returned. The value of *length2* can be no more than 153, which is the maximum length of a directory name.

actual_dirname_length
(output, INT, 4) is the actual length of the directory name returned in *dirname*. If no directory name is returned, it is zero.

type
(output, CHAR, 1) is a code indicating how the file ID compound parameter was specified:

1
A file mode letter

2
An SFS directory ID (except a file mode letter) or a BFS identification

3
A namedef.

Usage Notes

For a file in a directory, *actual_dirname_length* is greater than zero; it is zero otherwise. This is a simple check for determining whether *namedef2* refers to a directory.

Return Codes and Reason Codes

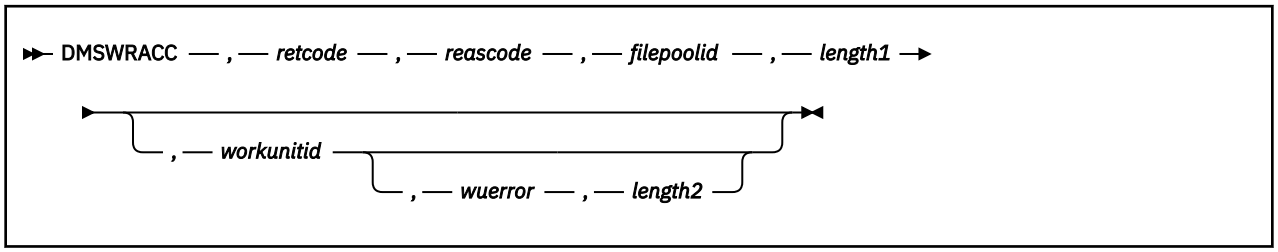
For lists of the possible return codes from DMSVALDT, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSVALDT. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSVALDT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	90700	Input in *dirid*, *filemode*, or *namedef2* was an unaccessed file mode.
ERROR	90270	The returned directory name is longer than the length specified in *length2*.
ERROR	90350	The compound parameter for the file ID is incorrect. There must be two, three, or four blank-delimited tokens in the string. Namedefs are treated as single strings.
ERROR	90410	Invalid parameter length specified.
ERROR	90420	The file name is incorrect. The file name is longer than eight characters or contains an incorrect character.
ERROR	90430	The file type is incorrect. The file type is longer than eight characters or contains an invalid character.
ERROR	90440	The specified file mode number is incorrect. It must be a single digit between '0' and '6'.
ERROR	90450	There are invalid characters (* or %) in *fn_ft*.
ERROR	90500	The specified directory name is incorrect. If the +/-*filemode.dirid* form of directory ID was specified, the file mode is not accessed or refers to a minidisk.
ERROR	90510	A namedef is longer than 16 characters.
ERROR	90530	There is an error in the file ID compound parameter. Some possible errors are: • A namedef does not exist or was used incorrectly. • The parameter does not resolve to a complete file ID; for example, the file mode was omitted.
ERROR	90590	There is no default file pool currently defined, and the file pool ID was not specified as part of the directory name.

DMSWRACC - Write File Pool Server Accounting Records



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSWRACC is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSWRACC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSWRACC routine to instruct a file pool server to write accounting records for all connected users. File pool administration authority is required.

For more information on file pool server accounting, see *z/VM: CMS File Pool Planning, Administration, and Operation*.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSWRACC.

reascode
(output, INT, 4) is a variable for the reason code from DMSWRACC.

filepoolid
(input, CHAR, 1-8) is a variable identifying the name of the file pool whose server accounting records are to be written. When specifying this parameter, an appended colon is invalid and should not be used.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*filepoolid*).

workunitid
(input, INT, 4) is a variable for specifying the work unit ID to be associated with this request. If you want to specify an optional parameter following *workunitid* without using the *workunitid* parameter, specify a value of 0 for the *workunitid* parameter.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify

a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and will result in data being overlaid.

Usage Notes

1. DMSWRACC is an atomic request. This means that there can be no outstanding work for the file pool on the specified work unit when this routine is called. When it is finished, DMSWRACC causes the work in the file pool to be committed without coordination.
2. The following defines **connected user**:
   - For a file pool server performing SFS or BFS repository functions, the connected user could be any user (on any processor) who is using the file pool.
   - For a file pool server performing FIFO server functions, the connected user could be any user (on any processor) who is doing FIFO operations through the server.
   - For a file pool server performing CRR recovery server functions, the connected user could be any user on the same processor who uses (at any point in the logon session) a CRR participating resource, such as SFS or a protected APPC conversation. (These are CRR logging connections.)

Return Codes and Reason Codes

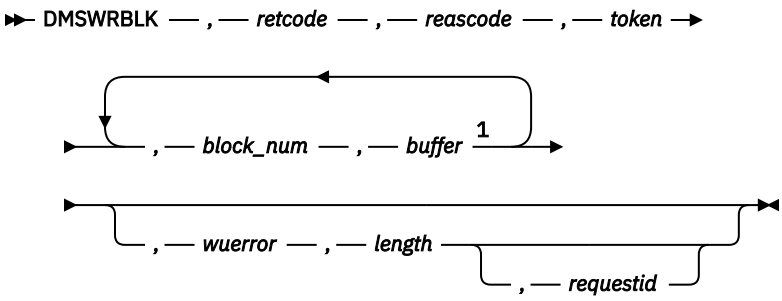
For lists of the possible return codes from DMSWRACC, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSWRACC. ERROR indicates that the request failed. Errors cause return codes 8 or 12.

DMSWRACC can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
ERROR	30000	You do not have file pool administration authority.
ERROR	30200	NOACCOUNT specified as a file pool server start-up parameter.
ERROR	76000	System error in file pool server commit function. The current unit of work has been rolled back.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90540	Specified work unit ID is invalid.
ERROR	95400	A logical unit of work is already in process for this file pool for the specified work unit ID.

DMSWRBLK - Write Blocks



Notes:

[1] Up to 7 block numbers and buffers can be specified.

Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSWRBLK is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSWRBLK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSWRBLK routine to write blocks to a file that has been opened using the DMSOPBLK (Open Blocks) routine.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSWRBLK.

reascode
(output, INT, 4) is a variable for the reason code from DMSWRBLK.

token
(input, CHAR, 8) contains a token which was returned on DMSOPBLK for this file.

block_num
(input, INT, 4) is the block number to be written. Up to seven block numbers can be specified. A block number of 0 indicates the end of the list of block numbers to be written. All following block numbers are ignored. However, all seven block numbers and buffers must be specified if you specify the work unit error parameter.

buffer
(input, CHAR, 4K) is an area from which the 4KB block will be written. Up to seven buffers can be specified. If the first block number specified is block 5, the first buffer will contain the 4 K bytes of data of block 5. If the n^{th} block number is 0, then the corresponding n^{th} buffer and all subsequent buffers are ignored.

wuerror
(output, CHAR, *length*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length
(input, INT, 4) is a variable containing the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes error reason code 90415 to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. Blocks can be written in any order.
2. The number of block numbers must equal the number of buffers. The block numbers and buffer areas are treated as pairs.
3. If the n^{th} block number is nonzero, 4KB of data will be written from the corresponding buffer.
4. A request ID is returned if the request is to be asynchronous.
5. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
6. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
7. If an I/O error occurs while you are writing blocks, execute the Release Blocks (DMSRELBK) routine. The Release Blocks routine releases alternate blocks that may be held in the file pool after an I/O error occurs using Write Blocks.
8. Specifying a block number more than once in the same request results in an error reason code from CMS.

Return Codes and Reason Codes

For lists of the possible return codes from DMSWRBLK, see Appendix D, “Return Codes,” on page 597.

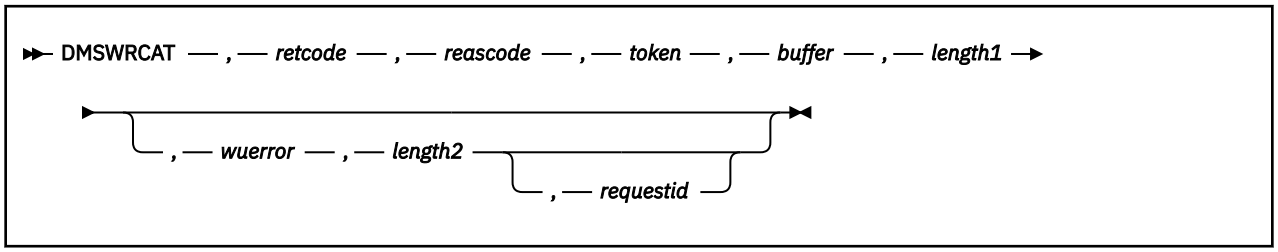
The following table lists the special reason codes returned by DMSWRBLK. WARNING means the request was processed and there were some exceptional conditions, or the desired state already exists. ERROR indicates that the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSWRBLK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	51050	The file-space-warning threshold was reached or exceeded.
ERROR	10000	File is not open with intent NEW, WRITE, or REPLACE.
ERROR	51000	Storage group space limit exceeded.
ERROR	71200	Either SFS made an error in accessing the file, or you specified the same block number more than once.
ERROR	78105	Incorrect block number passed.

Severity	Reason Code	Description
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID specified; must be 0 or 1.
ERROR	90488	Incorrect number of buffers specified.
ERROR	90490	Incorrect number of blocks specified.
ERROR	95700	System error. No open file pool object found for the specified token.

DMSWRCAT - Write Catalog



Context

File Pool Administration

Call Format

The format for calling a CSL routine is language dependent. DMSWRCAT is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSWRCAT
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSWRCAT routine to write catalog information to the storage group catalogs that were opened using the DMSOPCAT (Open Catalog) routine. File pool administration authority is required.

Use this routine only to restore catalog information. Any information written to the catalogs must have been derived from information previously obtained by the DMSRDCAT (Read Catalog) routine. If you are coding your own backup/restore program for SFS files, you should back up the files by using the block routines to copy the file blocks to some other media. At that time, you should also copy the related catalog information.

Restore the files by locking the objects to be restored, using the block routines to put back the files, and using this routine to restore the catalog information.

Note that some analysis of catalog information that currently exists in the file pool may be necessary before you restore a portion of the catalog data. For example, aliases may exist on the file that didn't exist at the time of the backup, authorizations may have been added or deleted, the user may no longer be enrolled, and so on. You would need to make decisions about how to handle these conditions and adjust the catalog information accordingly.

Remember that the Block routines and Catalog routines are low-level routines. They give you the opportunity for a better-performing backup/restore program, but they also give you the opportunity to damage files and to corrupt the file pool catalogs.

Unless your application demands optimal performance, consider using higher-level CSL routines or CMS commands instead of the block and catalog routines. For example, some of the techniques used in the TRANSFER exec in the *z/VM: CP Planning and Administration* could be used in a backup/restore program to manipulate the catalogs at a higher (and less dangerous) level.

If you use the block and catalog routines, do not change the internal object identifiers when you are restoring the catalogs. The internal identifiers uniquely identify file pool objects. Also, because of the risk of corrupting the file pool catalogs, it is recommended that you do extensive testing on a test file pool before using your program on a production file pool.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSWRCAT.

reascode
(output, INT, 4) is a variable for the reason code from DMSWRCAT.

token
(input, CHAR, 8) contains a token which was returned when the file was opened by the Open Catalog routine (DMSOPCAT).

buffer
(input, CHAR, *length1*) is a variable containing the catalog information for the Write Catalog request. It is in the format used in a previous Read Catalog request (DMSRDCAT), where the first fullword contains the length of the data. This is followed by the catalog information.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for CMS to return extended error information. If it is specified, it must be followed by a length field.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the effect of omitting the wuerror parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes an error reason code to be returned.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is synchronous. If it contains a binary 1 on input, the request is asynchronous and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check request. If, on return, the *requestid* is still 1, no server call was needed. It will not be necessary to call DMSCHECK because the function has already been completed.

Usage Notes

1. This routine is valid only for storage groups.
2. It is recommended that the buffer length be a multiple of 4KB.
3. A request ID is returned if the request is to be asynchronous.
4. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
5. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.

Return Codes and Reason Codes

For lists of the possible return codes from DMSWRCAT, see Appendix D, "Return Codes," on page 597.

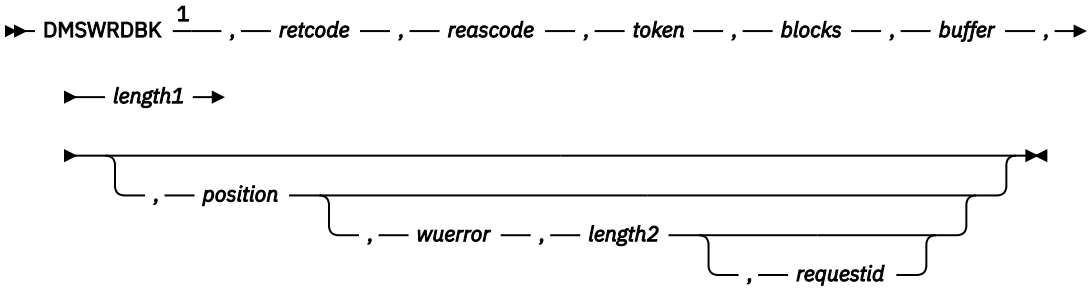
The following table lists the special reason codes returned by DMSWRCAT. ERROR indicates that the request failed. Errors cause return codes 8 or 12.

DMSWRCAT can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	56300	Invalid record type.

Severity	Reason Code	Description
ERROR	56400	Catalog is not open, or the requested operation conflicts with the open intent.
ERROR	90215	Invalid buffer length.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid requestid specified, must be 0 or 1.
ERROR	95700	No open file pool object found for the specified token.

DMSWRDBK - Write Data Block



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for invoking a CSL routine is language dependent. This CSL routine can be called directly by its name, DMSWRDBK, or called indirectly through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSWRDBK
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSWRDBK routine to write data blocks to a file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSWRDBK.

reascode
(output, INT, 4) is a variable for the reason code from DMSWRDBK.

token
(input, CHAR, 8) is a token that uniquely identifies a file to be written to. It was returned from a previous Open Data Block (DMSOPDBK).

blocks
(input, INT, 4) is the number of blocks to be written.

buffer
(input, CHAR, *length1*) is the area containing the data to be written. The declared length of this variable must correspond to the value you specify in the *length1* parameter.

length1
(input, INT, 4) is the length of the preceding character parameter (*buffer*).

position
(input, INT, 4) is the block number of the first block to be written, relative to the beginning of the file (block 1). If the position is not specified, or 0 is specified, the next sequential block is written.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is the length of the preceding character parameter (*wuerror*). Specifying 0 has the same effect as omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and will result in an error return code.

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

The processing of a given DMSWRDBK request may not require communication with the file pool server. In this case, the operation will be performed synchronously regardless of the value specified in *requestid*, and the value of the *requestid* parameter will not be modified by the operation.

Usage Notes

1. DMSWRDBK updates the write pointer so that, if the *position* parameter is specified as 0 (or is omitted) on the next DMSWRDBK operation, writing begins following the last block written by this DMSWRDBK.
2. For a new file, writing begins with block 1 unless specified otherwise with the *position* parameter.
3. For files with fixed-length records only, it is permissible to write a block with a position number more than one greater than the number of the last block. If the skipped blocks are not written before the file is closed, the file is termed a *sparse* file. A block may be written with a previously skipped position number when the file is subsequently reopened. If an attempt is made to read a block that has never been written, it is retrieved as all X'00' bytes.
4. DMSWRDBK fails if the target file is already open for READ or if the file was not opened with a call to DMSOPDBK.
5. Updates to a file with the INPLACE attribute may be only partially completed in the event of an abnormal termination.
6. When working with files that have the NORECOVER attribute:
   - Writes are shadowed, but are not subject to application initiated rollback.
   - Updates tend to be committed on application-initiated rollback.
   - Writes must be committed to guarantee they get written to DASD.
7. The buffer passed on the DMSWRDBK request should be neither examined nor modified while the asynchronous request is pending. Otherwise, inconsistent results may be obtained.
8. Data cannot be committed while this file is opened for data block I/O. The file must be closed (DMSCLDBK) for data to be committed.
9. For an asynchronous request, a return code is given indicating whether the request was accepted for processing or was immediately rejected.
10. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the affected file pool on the specified work unit.
11. All minidisk requests are done synchronously.

Return Codes and Reason Codes

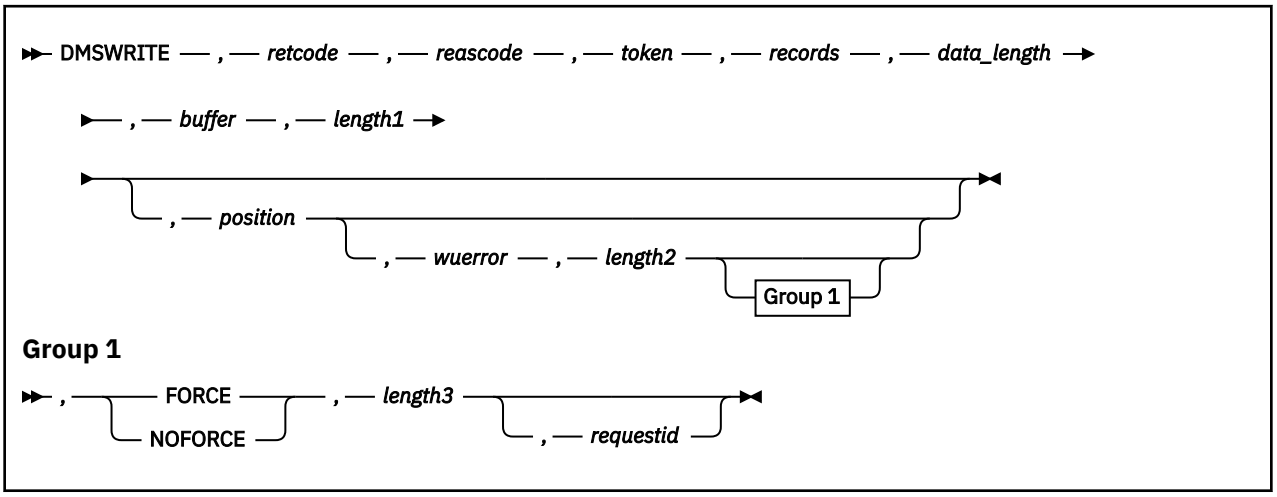
For lists of the possible return codes from DMSWRDBK, see Appendix D, “Return Codes,” on page 597.

The following table lists the special reason codes returned by DMSWRDBK. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

DMSWRDBK can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see “Common Reason Codes” on page 601.

Severity	Reason Code	Description
WARNING	51050	File space warning threshold reached or exceeded.
ERROR	10000	File is not open with intent NEW or REPLACE.
ERROR	30700	System error. An attempt was made to update a block in place for a file that is not OPEN for INPLACE processing.
ERROR	51000	Storage group space limit exceeded.
ERROR	54000	System error. Attempt to read logical block number that is not associated with the file. This can occur on a write request due to CMS buffering.
ERROR	90108	Size of input or output buffer is not greater than zero or an attempt to write a null record to a file having variable-length records.
ERROR	90111	Invalid buffer address.
ERROR	90112	Position specifies a negative block number.
ERROR	90113	Position plus the number of blocks to write exceeds 2^{31} - 1, the file system capacity.
ERROR	90114	In a variable format file, the position of the write pointer is neither at zero nor at the start of the next block.
ERROR	90129	File system capacity exceeded: the design limit does not permit more than 2^{31} - 1 blocks to be allocated to the file.
ERROR	90131	Insufficient minidisk space available.
ERROR	90146	Nonzero bytes follow end of data.
ERROR	90180	Buffer size not equal to the number of blocks multiplied by the block size.
ERROR	90415	Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Invalid request ID specified, must be 0 or 1.
ERROR	90490	Invalid number of blocks specified.
ERROR	95700	System error. No open file found for internal token passed to SFS.
ERROR	95750	No file opened using DMSOPDBK found for the specified token.

DMSWRITE - Write



Context

File System Management: SFS, BFS, and minidisk

Call Format

The format for calling a CSL routine is language dependent. DMSWRITE is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSWRITE
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSWRITE routine to write records to a file.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSWRITE.

reascode
(output, INT, 4) is a variable for the reason code from DMSWRITE.

token
(input, CHAR, 8) is a variable for passing the information that uniquely identifies a file to be written to. It was returned by the call to the Open routine (DMSOPEN) that opened the file for writing.

records
(input, INT, 4) is a variable for specifying the number of records to be written. For a file containing variable-length records this must be a 1.

data_length
(input, INT, 4) is a variable for specifying the size in bytes of the data to be written. For a file with variable-length records, this parameter specifies the length of the record to be written. For a file with fixed-length records, this parameter must be equal to the product of the *records* parameter and the logical record length. This must be a positive signed integer variable.

buffer
(input, CHAR, *length1*) is a variable for specifying the area containing the data to be written. The declared length of this variable must correspond to the value you specify in the *length1* parameter.

length1
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*buffer*).

position
(input, INT, 4) is a variable for specifying the record number of the first record to be written, relative to the beginning of the file (record 1). If the position is not specified, or 0 is specified, the next sequential record is written.

wuerror
(output, CHAR, *length2*) is a variable used to specify an area for returning extended SFS error information. If it is specified, it must be followed by a length field. See *wuerror* under "Common Parameters" on page 15 for more information.

length2
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (*wuerror*). Specifying 0 has the same effect as omitting the *wuerror* parameter. To use the *wuerror* parameter, specify a length of 12 plus 136 bytes for each group of extended error information that may be passed back. Specifying a nonzero length less than 12 is incorrect and causes error reason code 90415 to be returned.

FORCE
(input, CHAR, 5) causes this DMSWRITE operation to transmit all updates made by this request (and pending, uncommitted DMSWRITE requests) to the Shared File System. This enables readers to access INPLACE data as it is written to a file by a single concurrent writer. Omitting this parameter is the same as specifying NOFORCE.

FORCE enables an application to specify explicitly when updates to INPLACE files are to be transmitted to the file pool. Some updates to INPLACE files will then be made permanent and visible to other readers. The FORCE parameter has no effect when:

- The file's recoverability attribute is RECOVER
- Or the file was opened with the OPENRECOVER option
- Or the file is on a minidisk.

Omitting this parameter is the same as specifying NOFORCE.

NOFORCE
(input, CHAR, 7) lets the file system optimize the performance of this DMSWRITE operation by using data in CMS's file system buffers where possible.

When you use the NOFORCE option, the timing of the transfer of data to the Shared File System is not controlled by the application.

When NOFORCE is specified (or implied) and the file has the INPLACE attribute, the writer cannot differentiate between the updates that are visible to concurrent readers and those that are inaccessible until after a commit.

When a file has the INPLACE attribute and there are no concurrent readers, a DMSWRITE with the NOFORCE option will perform better and generate results similar to a DMSWRITE with the FORCE option.

When a file has the INPLACE attribute and there is one writer and more concurrent readers, NOFORCE processing by the writer may result in obsolete data being retrieved by the reader. This is because there may be some records that have been modified in the CMS file system buffers rather than directly into the Shared File System.

Omitting this parameter is the same as specifying NOFORCE.

length3
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (FORCE or NOFORCE).

requestid
(input/output, INT, 4) identifies a specific asynchronous request. If it is omitted or contains a binary 0 on input, the request is to be synchronous. If it contains a binary 1 on input, then the request is to be asynchronous, and CMS generates an integer to identify the asynchronous request. This integer is placed in *requestid*, which is passed on a later Check (DMSCHECK) request.

The processing of a given DMSWRITE request may not require communication with the file pool server. In this case, the operation will be performed synchronously regardless of the value specified in *requestid*, and the value of the *requestid* parameter will not be modified by the operation.

Usage Notes

1. DMSWRITE updates the write pointer so that, if the *position* parameter is specified as 0 (or is omitted) on the next DMSWRITE operation, writing begins following the last record written by this DMSWRITE.
2. For a new file, writing begins with record 1 unless specified otherwise with the *position* parameter. For existing files, writing begins following the last record in the file unless the *position* parameter is specified with a nonzero value to indicate the number of the record to be written. Current position is not affected by a commit request You can also use the DMSPOINT function to alter the write pointer in the file.
3. For files with fixed-length records only, it is permissible to write a record with a position number more than one greater than the number of the last record. If the skipped records are not written before the file is closed, the file is termed a *sparse* file. A record may be written with a previously skipped position number when the file is subsequently reopened. If an attempt is made to read a record that has never been written, it is retrieved as all X'00' bytes.
4. Variable-length records can be up to 65,535 bytes long. If you use DMSWRITE to update an existing file of variable-length records, the replacement record must be the same length as the original record. If it is not, the write fails and reason code 90121 is returned.
5. Null (zero-length) variable-length records are not supported.
6. DMSWRITE fails if the target file is already open for READ or if the file was not opened with a call to DMSOPEN.
7. Updates to a file with the INPLACE attribute may be only partially completed in the event of an abnormal termination.
8. When working with files that have the NORECOVER attribute:
   - Writes are shadowed, but are not subject to application initiated rollback.
   - Updates tend to be committed on application-initiated rollback.
   - Writes must be committed to guarantee they get written to DASD.
9. Remember the following rules when you use the FORCE option to write to INPLACE SFS files:
   - Records written past the end of file (new records, in other words) are not available to other readers until the work unit is committed. Similarly, newly created files are not available to other readers until the work unit is committed.
   - Files that are opened for REPLACE are not updated in place until after the first commit request. A reader who opens the file prior to that point continues to see the previous version of the file.
   - Updates to blocks that previously were sparse are not available to any reader until after the work unit is committed.

   In these three cases, if a call to DMSWRITE specifies the FORCE option, a warning is issued. For a reader to see these updates, the writer must issue a commit request and the reader must reopen the file.

   This option is designed to be used with the REFRESH option on DMSREAD. Otherwise, the data read by a concurrent reader may reflect updates at unpredictable intervals.

   The following return codes have the following meanings if you have specified the FORCE option:

- Return code 0 indicates a successful completion. All data has been updated in place. This implies that the updates have been committed and are available to existing readers that issue a DMSREAD with the REFRESH option and have the file open at the current committed level of the file.
- A warning (return code 4) is returned if the FORCE option was specified and the write operation was successfully completed, but some of the updates were not made permanent. This return code is returned when FORCE is specified on a DMSWRITE request and one or more of the following conditions are true:

  a. The file has the NOTINPLACE attribute.

  b. This DMSWRITE operation caused new records to be appended to the file.

  c. This DMSWRITE operation caused new blocks to be allocated to the file. This can occur either because blocks were appended to the file, or because file blocks that previously contained only zeros now have nonzero data in them.

  d. A previous uncommitted DMSWRITE operation caused condition "9.b" on page 550 or "9.c" on page 550 to occur.

  The updates may have been sent to the server, but are not committed. Some or all of updates will not be available to existing readers and in the event of an abnormal termination, the updates could be rolled back.
- An error (return code 8 or 12) will occur if any part of the data cannot be transmitted successfully to the server. If the file has the INPLACE attribute, a subset of the data may be committed to the file.

10. Because specifying FORCE may cause considerable performance overhead, it is suggested that you use it with caution. If you are designing applications that will use the FORCE option, one of the following approaches is suggested:
    - Make the record length large and unlikely to span 4KB blocks (for example, a multiple of 4KB).
    - Make the application buffer records before issuing the DMSWRITE.
    - Specify FORCE only on selected DMSWRITE operations.
11. The buffer passed on the DMSWRITE request should be neither examined nor modified while the asynchronous request is pending. Otherwise, inconsistent results may be obtained.
12. Changes cannot be committed if there is an asynchronous write operation pending. If the file is non-recoverable, and an asynchronous write operation is pending, data updated since the last commit operation may be lost if the work unit is rolled back.
13. A return code of 0 or 4 from an asynchronous request indicates only that the request was accepted for processing; processing errors can still occur. A return code of 0 or 4 from a synchronous request indicates that the operation was completed successfully.
14. If you have an active asynchronous request for a file pool, you cannot issue any other requests (except a rollback request) for the file pool on the same work unit.
15. Requests for minidisks are always processed synchronously and *requestid* is returned unchanged.

Return Codes and Reason Codes

For lists of the possible return codes from DMSWRITE, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSWRITE. WARNING means the request was processed, or the desired state already exists. ERROR means the request failed. Warnings cause return code 4, and errors cause return code 8 or 12.

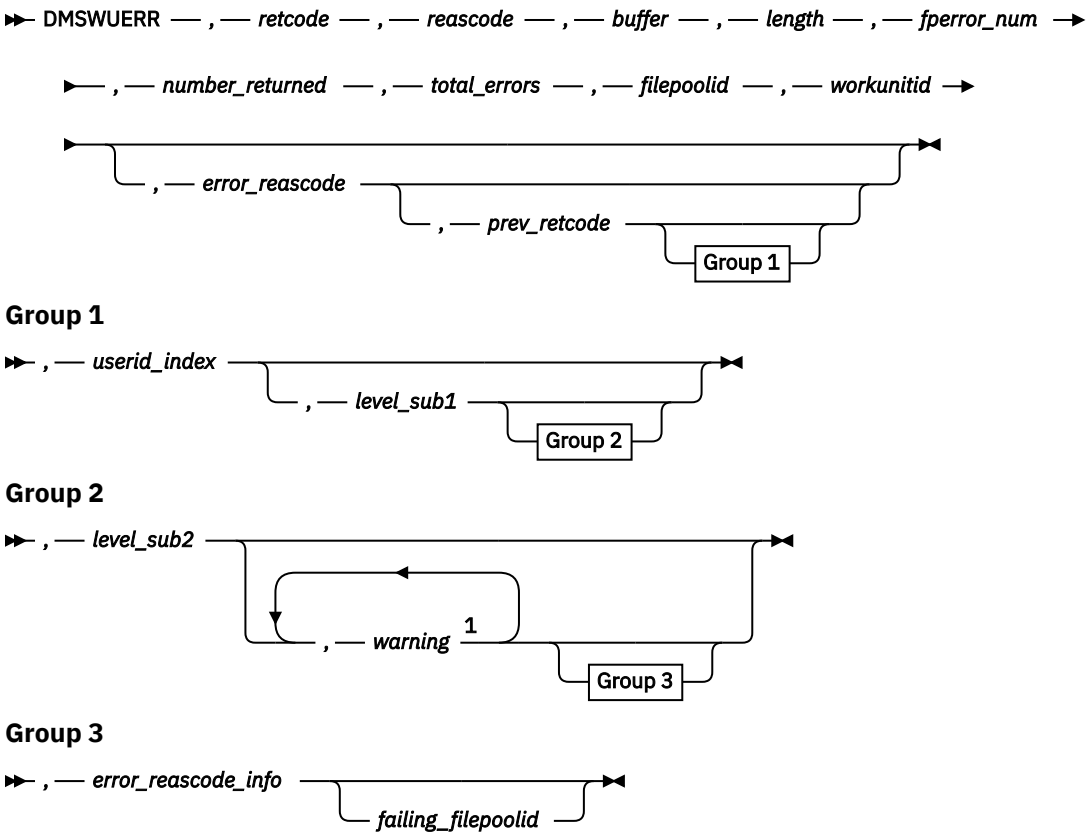
DMSWRITE can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
WARNING	51050	The file-space-warning threshold was reached or exceeded.

Severity	Reason Code	Description
WARNING	90144	The FORCE option was specified and some or all file updates are not permanent.
WARNING	90145	The FORCE option was specified, some or all file updates are not permanent and file space warning threshold reached or exceeded.
ERROR	10000	File is not open with intent NEW, WRITE, or REPLACE.
ERROR	30700	System error. An attempt was made to update a block in place for a file that is not OPEN for INPLACE processing.
ERROR	51000	Storage group space limit exceeded.
ERROR	54000	System error. Attempt to read logical block number that is not associated with the file. This can occur on a write request due to CMS buffering.
ERROR	90105	Incorrect record format.
ERROR	90106	Number of records to write is not greater than zero.
ERROR	90107	Number of records to write is not exactly one for a file containing variable-length records.
ERROR	90108	Size of input buffer is not greater than zero, or you have attempted to write a null record to a file containing variable-length records.
ERROR	90109	Specified data length is not evenly divisible by the number of records to be written to a file containing fixed-length records.
ERROR	90110	Size of input buffer is greater than 65535 for a file containing variable length records.
ERROR	90111	Incorrect buffer address.
ERROR	90112	Position specifies a negative record number.
ERROR	90113	Position plus the number of records to write exceeds 2^{31} - 1, the file system capacity.
ERROR	90114	Position specifies a record number that is more than one greater than the current number of records in a file containing variable length records.
ERROR	90115	File system capacity exceeded; a write operation attempted to put more than 2^{31} - 1 times blocksize bytes of data in a file, requiring a logical block number greater than 2^{31} - 1. This can occur for files having either fixed-length records or variable-length records: • For a file with fixed-length records, it occurs when the product of the record number of the last record to be written and the logical record length is greater than 2^{31} - 1 times the block size. • For a file with variable-length records, it occurs when the sum of the byte offset of a variable-length record and the length of that record is greater than 2^{31} - 1 times the block size.
ERROR	90120	You have attempted to alter the record length of a file containing fixed length records.

Severity	Reason Code	Description
ERROR	90121	You have attempted to replace an existing variable-length record with one of a different length.
ERROR	90129	File system capacity exceeded: the design limit does not permit more than 2^{31} - 1 blocks to be allocated to the file.
ERROR	90131	Insufficient minidisk space for a write operation to a minidisk file.
ERROR	90283	Data length specified exceeds the size of the input buffer.
ERROR	90310	Incorrect option in CSL parameter list - must be FORCE or NOFORCE.
ERROR	90320	Conflicting options in CSL parameter list.
ERROR	90330	Duplicate options in CSL parameter list.
ERROR	90415	Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.
ERROR	90472	Incorrect request ID: must be 0 or 1.
ERROR	95700	System error. No open file found for internal token passed to SFS.
ERROR	95750	No file opened using DMSOPEN found for the specified token.

DMSWUERR - Work Unit Error Data Deblocker



Notes:

[1] Up to 16 warnings can be specified.

Call Format

The format for calling a CSL routine is language dependent. DMSWUERR is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DMSWUERR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DMSWUERR routine to convert work unit error (*wuerror*) output from the structured format returned in other routines to data placed in individual variables.

Parameters

Note: See "Syntax Conventions for CSL Routines" on page 14.

retcode
(output, INT, 4) is a variable for the return code from DMSWUERR.

reascode
(output, INT, 4) is a variable for the reason code from DMSWUERR.

buffer
(input, CHAR, *length*) is the *wuerror* parameter returned on a previous call to another CSL routine. It contains the extended error information that is to be placed in specific variables.

length
(input, INT, 4) is a variable for specifying the length of the buffer area identified by the preceding character parameter (*buffer*).

fperror_num
(input, INT, 4) is a variable in which you specify the relative number of the extended error information you want returned. If multiple errors occurred, this number specifies the error for which information is to be returned for this call. Typically, this parameter should contain a value of 1 on the first call.

number_returned
(output, INT, 4) is a variable used to return the number of errors for which extended information was returned on a prior call to an SFS function.

total_errors
(output, INT, 4) is a variable used to return the total number of errors for which information was available. As many as would fit in the user-supplied area on a previous call were filled in. This number will be equal to or larger than the *number_returned* parameter.

filepoolid
(output, CHAR, 8) is a variable used to return the name of the file pool that was used on the previous call which found the error.

workunitid
(output, INT, 4) is a variable used to return the work unit used on the previous call which found the error.

error_reascode
(output, INT, 4) is a variable used to return the SFS error reason code associated with the set of extended error information being converted to individual variables. Warning reason codes are returned in the *warning* parameters; the *error_reascode* parameter contains a value of 0 when there are only warning reason codes.

prev_retcode
(output, INT, 4) is a variable used to hold the SFS error or warning return code associated with the set of extended error information being converted to individual variables.

userid_index
(output, INT, 4) is a variable used to return the index of the user ID for which an error or warning was first found for calls that have a list of user IDs. This value indicates the relative position of the user ID in the list.

level_sub1
(CHAR, 4) is a reserved field.

level_sub2
(CHAR, 4) is a reserved field.

warning
(output, INT, 4) is a variable used to hold a warning reason code. You can specify up to 16. All 16 must be specified if you use the error reason code information parameter.

error_reascode_info
(output, CHAR, 4) provides additional information for some *error_reascode* values:

71800
The *error_reascode_info* parameter contains the recovery token of the conflicting CRR resynchronization activity. Any display of this value should be hexadecimal.

If your application cannot access the required resource, it should inform the user that the resource is unavailable and provide the following information for the user to pass to the file pool administrator:

- The recovery token (returned in this parameter)

- The contents of the *filepoolid* parameter

50500
The *error_reascode_info* parameter contains the code for the reason the work unit could not be committed after a successfully completed operation. For explanations of these codes, see usage note "9" on page 72, "Return Codes and Reason Codes" on page 73, and "Common Reason Codes" on page 601.

failing_filepoolid
(output, CHAR, 8) is a variable used to hold the file pool ID of the failing resource. This will be significant only when a DFSMS/VM recall error has been encountered.

Usage Notes

1. The Work Unit Error Data Deblocker routine is used by languages that cannot handle the error information structure passed back in the *wuerror* parameter in other routines. The format passed back is described in *wuerror* under "Common Parameters" on page 15.
2. Information for one group of extended error information is returned each time DMSWUERR is called.
3. The *error_reascode* and *prev_retcode* parameters may not contain the same values as the *reascode* and *retcode* parameters returned on the CSL routine for which extended error information was generated. For example, if there was an attempt to commit a work unit, generic return and reason codes may be generated, in which case each set of extended error information in the *wuerror* buffer is used for the SFS-specific error information.
4. The *error_reascode_info* field information may contain a CSL error reason code returned from DFSMS/VM in the event of a failed recall attempt.

Return Codes and Reason Codes

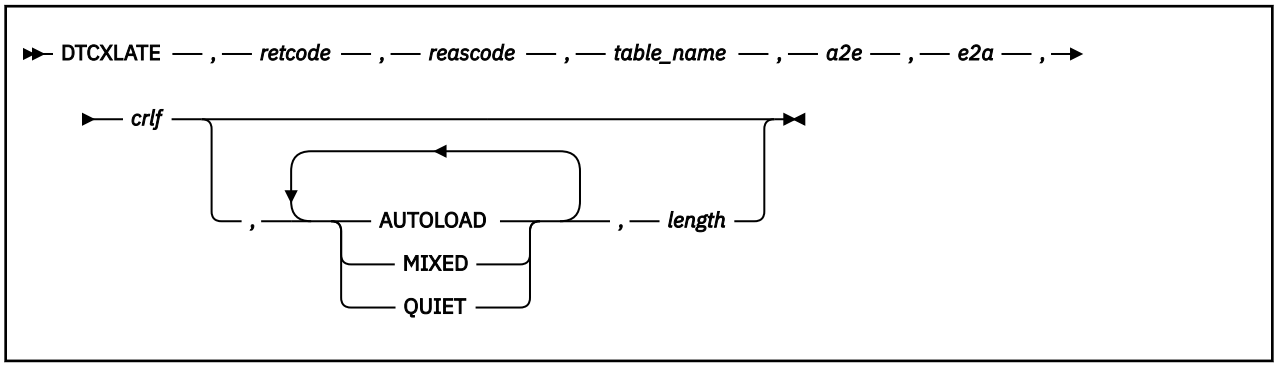
For lists of the possible return codes from DMSWUERR, see Appendix D, "Return Codes," on page 597.

The following table lists the special reason codes returned by DMSWUERR. ERROR indicates that the request failed. Errors cause return code 8 or 12.

DMSWUERR can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	76055	Invalid FPERROR number specified. It is less than 1 or greater than the number of FPERROR fields returned.
ERROR	76056	The *wuerror* buffer contained no FPERROR information.
ERROR	90215	Invalid buffer length specified for input *wuerror* buffer.

DTCXLATE - Read TCP/IP Translation Table



Call Format

The format for calling a CSL routine is language dependent. DTCXLATE is called only through DMSCSL. The routine name is the first parameter in DMSCSL's parameter list:

DTCXLATE
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the DTCXLATE routine to obtain translation information from a TCP/IP translation table file (TCPXLBIN file).

Parameters

retcode
(output, INT, 4) is a variable for the return code from DTCXLATE.

reascode
(output, INT, 4) is a variable for the reason code from DTCXLATE.

table_name
(input, CHAR, 8) is a variable for specifying the name of the TCP/IP translation table to be loaded.

The special value "*BUILTIN" can be specified to load the default 7-bit ASCII translation table. It is equivalent to STANDARD TCPXLBIN as shipped by IBM.

a2e
(output, CHAR, 256) is a variable to contain the 256-byte translation table to be used to convert ASCII to EBCDIC.

e2a
(output, CHAR, 256) is a variable to contain the 256-byte translation table to be used to convert EBCDIC to ASCII.

crlf
(output, CHAR, 2) is a variable to contain the EBCDIC codes that, when converted to ASCII using the *e2a* translation table, will generate ASCII carriage return (CR, X'0D') and line feed (LF, X'0A').

The EBCDIC equivalent of ASCII CR and LF characters is determined based on the particular translation table chosen.

AUTOLOAD
(input, CHAR, 8) indicates that STANDARD TCPXLBIN will be loaded if *table_name* cannot be loaded. If STANDARD cannot be loaded, then the default 7-bit translation table is loaded. This table is equivalent to STANDARD, as shipped by IBM.

MIXED
(input, CHAR, 5) indicates that *table_name* will not be translated to uppercase.

QUIET
(input, CHAR, 5) suppresses error messages.

length
(input, INT, 4) is a variable for specifying the length of the preceding character parameter (AUTOLOAD, MIXED, and QUIET). The maximum value for this parameter is 256. For details on coding, see "Compound Variables" on page 15.

Usage Notes

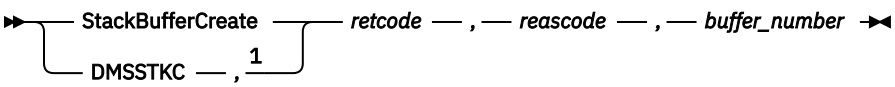
1. Only single byte character set (SBCS) translation tables can be used with this routine.
2. TCP/IP translation tables are created using the CONVXLAT command. For more information, see "Using Translation Tables" in *z/VM: TCP/IP Planning and Customization*.

Return Codes and Reason Codes

The following table lists the return codes and reason codes from DTCXLATE.

Return Code	Reason Code	Description
0	0	*table_name* was successfully loaded.
0	4	*table_name* was not found, but STANDARD TCPXLBIN has been loaded instead.
0	8	Neither *table_name* nor STANDARD TCPXLBIN could be loaded. The default 7-bit translation table has been loaded instead.
8	0	*table_name* was found, but could not be loaded because it is not a valid TCP/IP translation file.
8	28	*table_name* TCPXLBIN was not found
8	*n*	*table_name* TCPXLBIN could not be loaded due to an I/O error. The reason code contains the return code from the failing FSOPEN or FSREAD macro.
12	*n*	The address or value of parameter *n* is not valid. *retcode* is parameter number 1.

StackBufferCreate / DMSSTKC - Add a Buffer to the Program Stack



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Call Format

The format for invoking a CSL routine is language dependent. This routine can be called directly by either name or indirectly through DMSCSL as DMSSTKC. The routine name is the first parameter in DMSCSL's parameter list:

DMSSTKC
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the StackBufferCreate routine to add a buffer to the top of the program stack.

Parameters

retcode
(output, INT, 4) is a variable for the return code from StackBufferCreate.

reascode
(output, INT, 4) is a variable for the reason code from StackBufferCreate.

buffer_number
(output, INT, 4) is a variable to hold the number of the added buffer.

Usage Notes

1. StackBufferCreate provides the same function as the CMS MAKEBUF command. Calls to StackBufferCreate and MAKEBUF can be intermixed to add buffers cumulatively to the program stack. For more information on the program stack, see the *z/VM: CMS Application Development Guide*.
2. Buffers added to the program stack by StackBufferCreate can be read by the StackRead CSL routine (see "StackRead / DMSSTKR - Read from the Program Stack" on page 564) the LINERD macro, the WAITRD function, the PULL or PARSE REXX instruction, or the &READ statement in EXEC or EXEC 2.

Return Codes and Reason Codes

For lists of the possible return codes from StackBufferCreate, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by StackBufferCreate. ERROR means the request failed. The return code from successful requests is `vm_success`. Errors cause return code `vm_error`.

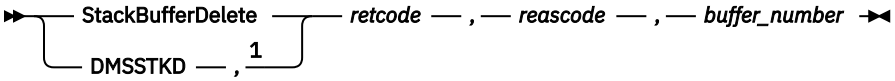
Severity	Reason Code	Description
ERROR	vm_stk_insufficient_storage	CMS was unable to obtain storage.

Programming Language Binding Files

Table 58. Binding Files for Function Definitions and Constants	
Language	**Binding File**
Assembler	VMASMSTK MACRO
C	VMCSTK H
COBOL	VMCOBSTK COPY
FORTRAN	VMFORSTK COPY
PASCAL	VMPASSTK COPY
PL/I	VMPLISTK COPY
REXX	VMREXSTK COPY

Table 59. Binding Files for Return Codes and Reason Codes	
Language	**Binding File**
Assembler	VMASMRET MACRO
C	VMCRET H
COBOL	VMCOBRET COPY
FORTRAN	VMFORRET COPY
PASCAL	VMPASRET COPY
PL/I	VMPLIRET COPY
REXX	VMREXRET COPY

StackBufferDelete / DMSSTKD - Drop Buffers from the Program Stack



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Call Format

The format for invoking a CSL routine is language dependent. This routine can be called directly by either name or indirectly through DMSCSL as DMSSTKD. The routine name is the first parameter in DMSCSL's parameter list:

DMSSTKD
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the StackBufferDelete routine to drop buffers from the program stack.

Parameters

retcode
(output, INT, 4) is a variable for the return code from StackBufferDelete.

reascode
(output, INT, 4) is a variable for the reason code from StackBufferDelete.

buffer_number
(input, INT, 4) is a variable to hold the number of the buffer to be dropped. Buffers with numbers higher than *buffer_number* are also dropped.

The buffer number can be a value from 0 to the highest current buffer number. When *buffer_number* is 0, the lines in buffer 0 are deleted and all buffers except buffer 0 are dropped.

Usage Notes

1. StackBufferDelete provides the same function as the CMS DROPBUF command. Calls to DROPBUF and StackBufferDelete can be intermixed.
2. Use StackQuery to find out the number of the top buffer (see "StackQuery / DMSSTKQ - Query the Program Stack" on page 562).

Return Codes and Reason Codes

For lists of the possible return codes from StackBufferDelete, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by StackBufferDelete. ERROR means the request failed. The return code from successful requests is `vm_success`. Errors cause return code `vm_error`.

Severity	Reason Code	Description
ERROR	vm_stk_bad_buffer_number	The buffer number is invalid.

Severity	Reason Code	Description
ERROR	vm_stk_no_buffer	There is no buffer with the specified buffer number.

Programming Language Binding Files

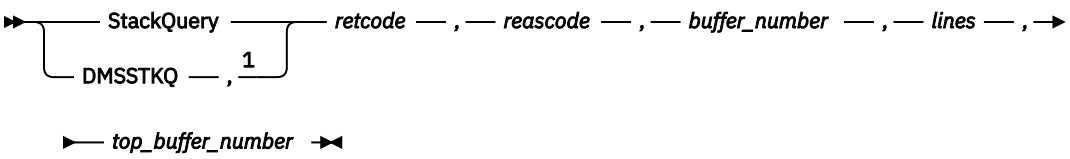
Table 60. Binding Files for Function Definitions and Constants

Language	Binding File
Assembler	VMASMSTK MACRO
C	VMCSTK H
COBOL	VMCOBSTK COPY
FORTRAN	VMFORSTK COPY
PASCAL	VMPASSTK COPY
PL/I	VMPLISTK COPY
REXX	VMREXSTK COPY

Table 61. Binding Files for Return Codes and Reason Codes

Language	Binding File
Assembler	VMASMRET MACRO
C	VMCRET H
COBOL	VMCOBRET COPY
FORTRAN	VMFORRET COPY
PASCAL	VMPASRET COPY
PL/I	VMPLIRET COPY
REXX	VMREXRET COPY

StackQuery / DMSSTKQ - Query the Program Stack



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Call Format

The format for invoking a CSL routine is language dependent. This routine can be called directly by either name or indirectly through DMSCSL as DMSSTKQ. The routine name is the first parameter in DMSCSL's parameter list:

DMSSTKQ
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the StackQuery routine to query the number of lines in the program stack or the number of lines in a specific stack buffer. The number of the top buffer in the program stack is also returned.

Parameters

retcode
(output, INT, 4) is a variable for the return code from StackQuery.

reascode
(output, INT, 4) is a variable for the reason code from StackQuery.

buffer_number
(input, INT, 4) is a value that indicates what data should be returned in the lines parameter. The buffer number parameter can be set to:

Value
Effect

0–*n*
The number of a particular buffer, 0 through the number of the top buffer in the program stack. StackQuery returns the number of lines in that buffer.

vm_stk_query_all_buffers
StackQuery returns the number of lines in the program stack.

vm_stk_query_top_buffer
StackQuery returns the number of lines in the top buffer of the program stack.

lines
(output, INT, 4) is a variable to hold the number of lines, as specified by *buffer_number*.

top_buffer_number
(output, INT, 4) is a variable to hold the number of the top buffer in the program stack.

Usage Notes

1. The top buffer number is always returned, even when the return code indicates an error.

2. StackQuery returns information about the program stack. It can replace calls to DMSERP to extract information (NUM_TERM_INPUT_LINES and NUM_FINISHED_READS) that is necessary to calculate the number of lines in the program stack.

Return Codes and Reason Codes

For lists of the possible return codes from StackQuery, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by StackQuery. ERROR means the request failed. The return code from successful requests is `vm_success`. Errors cause return code `vm_error`.

Severity	Reason Code	Description
ERROR	vm_stk_no_buffer	There is no buffer with the specified buffer number.
ERROR	vm_stk_bad_buffer_number	Invalid buffer number.

Programming Language Binding Files

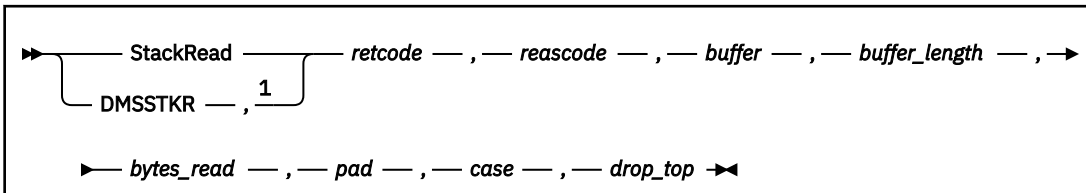
Table 62. Binding Files for Function Definitions and Constants

Language	Binding File
Assembler	VMASMSTK MACRO
C	VMCSTK H
COBOL	VMCOBSTK COPY
FORTRAN	VMFORSTK COPY
PASCAL	VMPASSTK COPY
PL/I	VMPLISTK COPY
REXX	VMREXSTK COPY

Table 63. Binding Files for Return Codes and Reason Codes

Language	Binding File
Assembler	VMASMRET MACRO
C	VMCRET H
COBOL	VMCOBRET COPY
FORTRAN	VMFORRET COPY
PASCAL	VMPASRET COPY
PL/I	VMPLIRET COPY
REXX	VMREXRET COPY

StackRead / DMSSTKR - Read from the Program Stack



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Call Format

The format for invoking a CSL routine is language dependent. This routine can be called directly by either name or indirectly through DMSCSL as DMSSTKR. The routine name is the first parameter in DMSCSL's parameter list:

DMSSTKR
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the StackRead routine to read a line from the program stack.

Parameters

retcode
(output, INT, 4) is a variable for the return code from StackRead.

reascode
(output, INT, 4) is a variable for the reason code from StackRead.

buffer
(output, CHAR, *buffer_length*) is a variable to hold the line read from the stack. If the line is shorter than the buffer, the buffer is padded as specified by the pad parameter. If the line is longer than the buffer, the line is truncated. The maximum length of a line on the stack is 255 bytes, but the buffer can be any length.

buffer_length
(input, INT, 4) is a variable for specifying the length of the buffer.

bytes_read
(output, INT, 4) is a variable that is set to the number of the bytes in the line read from the stack. If a null line is read or if there is an error condition, it is set to 0.

pad
(input, INT, 4) is a variable that indicates how to pad the line read from the stack to the length of the buffer. It must be set to one of these values:

Value
Effect

vm_stk_pad_blanks
Pad with blanks.

vm_stk_pad_nulls
Pad with nulls.

vm_stk_pad_none
Do not pad.

case
(input, INT, 4) is a variable indicating whether the returned line should be shifted to upper case. It must be set to one of these values:

Value
Effect

vm_stk_no_shift
Do not shift.

vm_stk_shift
Shift to upper case.

drop_top
(input, INT, 4) is a variable indicating what to do when the top buffer of the program stack is empty. It must be set to one of these values:

Value
Effect

vm_stk_drop_top
Drop empty top buffers until a line is found or the bottom of the stack is reached.

vm_stk_no_drop_top
Do not drop an empty top buffer.

Usage Notes

1. StackRead provides much of the same function as the CMS LINERD macro, the REXX PULL and PARSE PULL instructions, and the &READ instruction in EXEC and EXEC2. Calls to StackRead can be intermixed with calls to the other commands.
2. StackRead does not read from the virtual console or from the terminal input buffer.

Return Codes and Reason Codes

For lists of the possible return codes from StackRead, see Appendix D, “Return Codes,” on page 597.

The following table lists the reason codes returned by StackRead. ERROR means the request failed. The return code from successful requests is `vm_success`. Errors cause return code `vm_error`.

Severity	Reason Code	Description
ERROR	vm_stk_bad_buffer	The buffer address is incorrect.
ERROR	vm_stk_bad_buffer_length	The length of the buffer is not greater than or equal to zero.
ERROR	vm_stk_empty	The program stack is empty.
ERROR	vm_stk_top_buffer_empty	The top buffer of the program stack is empty.
ERROR	vm_stk_bad_case	Incorrect value for the case parameter.
ERROR	vm_stk_bad_pad	Incorrect value for the pad parameter.
ERROR	vm_stk_bad_drop_top	Incorrect value for *drop_top*.

Programming Language Binding Files

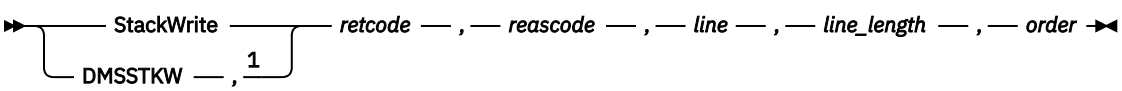
Table 64. Binding Files for Function Definitions and Constants

Language	Binding File
Assembler	VMASMSTK MACRO

Table 64. Binding Files for Function Definitions and Constants (continued)	
Language	**Binding File**
C	VMCSTK H
COBOL	VMCOBSTK COPY
FORTRAN	VMFORSTK COPY
PASCAL	VMPASSTK COPY
PL/I	VMPLISTK COPY
REXX	VMREXSTK COPY

Table 65. Binding Files for Return Codes and Reason Codes	
Language	**Binding File**
Assembler	VMASMRET MACRO
C	VMCRET H
COBOL	VMCOBRET COPY
FORTRAN	VMFORRET COPY
PASCAL	VMPASRET COPY
PL/I	VMPLIRET COPY
REXX	VMREXRET COPY

StackWrite / DMSSTKW - Write to the Program Stack



Notes:

[1] The comma is omitted after the routine name when the routine is called directly.

Call Format

The format for invoking a CSL routine is language dependent. This routine can be called directly by either name or indirectly through DMSCSL as DMSSTKW. The routine name is the first parameter in DMSCSL's parameter list:

DMSSTKW
(input, CHAR, 8) can be passed as a literal or in a variable.

For more information and examples of the call formats, see "Calling VMLIB CSL Routines" on page 2.

Purpose

Use the StackWrite routine to write a line to the top buffer of the program stack.

Parameters

retcode
(output, INT, 4) is a variable for the return code from StackWrite.

reascode
(output, INT, 4) is a variable for the reason code from StackWrite.

line
(input, CHAR, *line_length*) is a variable containing the line to be written to the buffer.

line_length
(input, INT, 4) is a variable containing the length of *line*. The line length must be in the range 0 to 255. If the line length is 0, then a null line is written.

order
(input, INT, 4) is a variable indicating how the line should be written to the buffer: LIFO or FIFO. The order parameter can be assigned one of two values:

Value
Effect

vm_stk_lifo
Pushes the line.

vm_stk_fifo
Queues the line.

Usage Notes

StackWrite provides the same function as the CMS CMSSTACK macro. Calls to CMSSTACK and StackWrite can be intermixed.

Return Codes and Reason Codes

For lists of the possible return codes from StackWrite, see Appendix D, "Return Codes," on page 597.

The following table lists the reason codes returned by StackWrite. ERROR means the request failed. The return code from successful requests is `vm_success`. Errors cause return code `vm_error`.

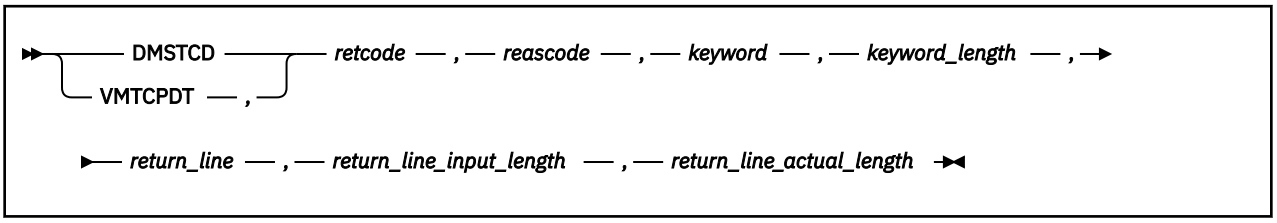
Severity	Reason Code	Description
ERROR	vm_stk_bad_line_length	The value of *line_length* was not in the range 0 to 255.
ERROR	vm_stk_insufficient_storage	CMS could not obtain storage. Nothing was placed on the program stack.
ERROR	vm_stk_bad_order	Invalid order parameter.

Programming Language Binding Files

Table 66. Binding Files for Function Definitions and Constants	
Language	**Binding File**
Assembler	VMASMSTK MACRO
C	VMCSTK H
COBOL	VMCOBSTK COPY
FORTRAN	VMFORSTK COPY
PASCAL	VMPASSTK COPY
PL/I	VMPLISTK COPY
REXX	VMREXSTK COPY

Table 67. Binding Files for Return Codes and Reason Codes	
Language	**Binding File**
Assembler	VMASMRET MACRO
C	VMCRET H
COBOL	VMCOBRET COPY
FORTRAN	VMFORRET COPY
PASCAL	VMPASRET COPY
PL/I	VMPLIRET COPY
REXX	VMREXRET COPY

VMTCPDT / DMSTCD - Parse TCPIP DATA File



Context

The format for calling a CSL routine is language dependent. This routine can be called directly (by module name) or indirectly (by CSL routine name):

DMSTCD
(input, CHAR, 6) is the name to be used when the routine is called directly.

VMTCPDT
(input, CHAR, 8) is the name to be used when the routine is called indirectly through DMSCSL. This name, which can be passed as a literal or in a variable, is the first parameter in DMSCSL's parameter list. "VMTCPDT" must be padded with blanks to eight characters.

For more information and examples of the call formats, see “Calling VMLIB CSL Routines” on page 2.

Purpose

Use the DMSTCD routine to provide an interface for parsing the TCPIP DATA file and returning the value associated with a specific requested keyword. The routine parses the first TCPIP DATA file in the search order of the user's virtual machine. It respects the system_name capability of the TCPIP DATA file, if used, and returns only information defined for this system. It does not validate the syntax or values within the TCPIP DATA file.

Parameters

retcode
(output, INT, 4) is a variable for the return code from DMSTCD. For more information on possible return codes, see the "Return Codes" section of this routine description.

reascode
(output, INT, 4) is a variable for the error reason code from DMSTCD. For more information on possible reason codes, see the "Reason Codes" section of this routine description.

keyword
(input, CHAR, *keyword_length*) is a variable for specifying the requested TCPIP DATA file parameter of interest.

keyword_length
(input, INT, 4) is a variable for specifying the length of the keyword parameter.

return_line
(output, CHAR, *return_line_input_length*) is the name of the buffer to contain the data portion of the TCPIP DATA file entry requested.

Comment portions of data lines (information from any ';'on) are not returned, only the token that is the data portion of the statement is returned. All leading and trailing blanks are removed.

return_line_input_length
(input, INT, 4) is the length of the return line buffer.

return_line_actual_length
(output, INT, 4) is the length of the returned data.

Usage Notes

1. The returned value is the token associated with the requested Keyword. The only exception to this is HOSTNAME. If HOSTNAME is the Keyword value and it is not found in TCPIP DATA (but TCPIP DATA exists), the system's local node name (as determined by the IDENTIFY command) is returned.
2. If the Keyword requested is found more than once in the TCPIP DATA file and is defined for this system (as NSINTERADDR might be, for example) then all tokens associated with this Keyword are returned in *return_line* Each value is returned as a separate token.

Return Codes and Reason Codes

The following table lists the special reason codes returned by DMSTCD. WARNING means the request was processed (RC=4). ERROR means the request failed (RC=8).

DMSTCD can also return the common CSL reason codes, which are codes that can occur with most file system management and related routines. For a list of the common reason codes, see "Common Reason Codes" on page 601.

Severity	Reason Code	Description
ERROR	96100	Insufficient virtual storage for a get storage request from user's virtual machine on error.
WARNING	90101	Data truncated. *return_line* was not big enough to hold the returned value.
WARNING	90220	TCPIP DATA file not found.
WARNING	99631	Keyword not found in TCPIP DATA file.

Appendix A. Using Programming Language Binding Files

Programs that use CSL routines with longer names like "StackBufferCreate", rather than short names like "DMSERP", need to include programming language binding files. Language binding files define the entry points; declare the data types of function parameters; map long function names to external symbols; and define constants for return codes, reason codes, and other values. The binding files are required even when these routines are called with DMSCSL using the alternative short names.

In the descriptions of these routines, the values that can be assigned to parameters and the values of return and reason codes are given by the symbols defined in the binding files. Each value has the same symbolic name in every language. These symbolic names are part of the formal API definition and should be used in programs instead of the actual numeric values. However, numeric values of reason and return codes can appear in error messages from other parts of the API, and Ada programs must use the numeric values. See the mapping of symbols to numeric values in Appendix B, “Symbols Defined in Binding Files,” on page 575.

Binding Files for Routines in This Book

z/VM supplies the following binding files for routines described in this book:

- For the stack services routines:
  - VMASMSTK and VMASMRET for assembler
  - VMCOBSTK and VMCOBRET for COBOL
  - VMCSTK and VMCRET for C
  - VMFORSTK and VMFORRET for FORTRAN
  - VMPASSTK and VMPASRET for Pascal
  - VMPLISTK and VMPLIRET for PL/I
  - VMREXSTK and VMREXRET for REXX
- For the WorkstationGetAddress routine:
  - VMASMWKS for assembler
  - VMCWKS for C and C++
  - VMREXWKS for REXX

Mechanisms for Including Binding Files

Table 68 on page 571 shows the mechanisms for including the binding files for different languages. This table and the accompanying usage notes use the binding files for the stack services routines as an example.

Table 68. Including Language Binding Files

Language	Location of Binding Files	Method
Ada	Binding files are not used.	Use shorter-style names to call routines and use numeric values for parameters (see Appendix B, “Symbols Defined in Binding Files,” on page 575).

Table 68. Including Language Binding Files (continued)

Language	Location of Binding Files	Method
Assembler	DMSGPI MACLIB	`CALL VMASMSTK` `CALL VMASMRET`
C	System disk	`#include <vmcstk.h>` `#include <vmcret.h>`
COBOL	DMSGPI MACLIB	`COPY VMCOBSTK` `COPY VMCOBRET`
FORTRAN	DMSGPI MACLIB	`INCLUDE VMFORSTK` `INCLUDE VMFORRET`
Pascal	DMSGPI MACLIB	`%INCLUDE VMPASSTK` `%INCLUDE VMPASRET`
PL/I	DMSGPI MACLIB	`% INCLUDE SYSLIB(VMPLISTK);` `% INCLUDE SYSLIB(VMPLIRET);`
REXX	System disk	`CALL APILOAD 'VMREXSTK'` `CALL APILOAD 'VMREXRET'`

Table 69. Usage Notes for Binding Files

Language	Notes
General	1. See the language guide for details of compiling, linking, and executing programs on z/VM. 2. See the *z/VM: CMS Application Development Guide* for sample programs in using CSL routines and examples of how to invoke them. 3. Before you compile your program, access the maclib that contains the binding file with the CMS GLOBAL MACLIB command. 4. When a program calls routines indirectly through DMSCSL, it must declare DMSCSL as an external routine. However, for direct calls, the binding files contain the external declarations for the routines.
C	1. Because C is case sensitive, the routine names must be coded exactly as they are declared in the `#pragma map` statement in VMCSTK H.

Table 69. Usage Notes for Binding Files (continued)

Language	Notes
COBOL	1. When you are using binding files, compile COBOL programs with the LIB and RESIDENT options. 2. Place the binding files in the working section of the program. 3. Because COBOL does not permit the underscore character in user-defined symbols, symbols defined in the COBOL binding files use hyphens instead of the underscores shown in the routine descriptions. For example, *vm-stk-lifo* is defined in VMCOBSTK COPY instead of *vm_stk_lifo*. 4. Do not use quotation marks on calls that use the longer names: `CALL StackQuery USING…`
FORTRAN	1. Routines must be invoked with the shorter-style names.
Pascal	1. Declare routines with the GENERIC directive. 2. Routines must be invoked with the shorter-style names. 3. Pass parameters by reference, as variables, rather than as literals. 4. Use the compiler option `LIB(DMSGPI)`. 5. When you generate a load module for a program that uses direct call CSL routines, specify VMLIB on the PASCMOD command to make the TEXT files for the routines available. 6. VMPASSTK COPY generates executable code.
PL/I	1. Routines must be invoked with the shorter-style names.

Assembler Programs

Assembler binding files include DSECTs that map the parameter list address block defined by type-1 linkage. The parameter list can be accessed either statically to produce serially reusable code, or dynamically to produce reentrant code. The first two calling techniques described here do not produce reentrant code; the third technique does. For example, StackQuery can be called this way:

```
         L     R1,address of data area
         USING VM_STKQ_PLIST,R1
         MVC   VM_STKQ_PLIST_RETCODE,=A(RC)
         MVC   VM_STKQ_PLIST_REASCODE,=A(RE)
         MVC   VM_STKQ_PLIST_BUFFER_NUMBER,=A(BUFNUM)
         MVC   VM_STKQ_PLIST_LINES,=A(LINES)
         MVC   VM_STKQ_PLIST_HIGHEST_BUFFER_NUMBER,=A(HIGH+X'80000000')
         L     R15,=A(STACKQUERY)
         BALR  R14,R15
```

Or, using the CALL macro, StackQuery can be called this way:

```
    L     R15,=A(STACKQUERY)
    CALL  (15),(RC,RE,BUFNUM,LINES,HIGH),VL
```

When the long form of a routine's name is used in a direct call, the routine's address must be passed in a register.

These two calling techniques produce serially reusable code, because the addresses of the parameters (`RC`, `RE`, `BUFNUM`, `LINES`, and `HIGH`) are constants computed at load time (note the use of "=A" in referring to those parameters). For reentrant code, all parameters and the parameter list must reside in dynamically obtained storage, and the parameter list must be built at run time using LA and ST instructions. For example, the following code fragment produces reentrant code, assuming:

- R1 points to an area of dynamically obtained storage for the parameter list
- There is a base register for dynamically obtained storage for the parameters themselves.

```
     USING    VMSTKQ_PLIST,R1
     LA       R2,RC
     ST       R2,VMSTKQ_PLIST_RETCODE
     LA       R2,RE
     ST       R2,VMSTKQ_PLIST_REASCODE
     LA       R2,BUFNUM
     ST       R2,VMSTKQ_PLIST_BUFFER_NUMBER
     LA       R2,LINES
     ST       R2,VMSTKQ_PLIST_LINES
     LA       R2,HIGH
     ST       R2,VMSTKQ_PLIST_HIGHEST_BUFFER_NUMBER
     OI       VMSTKQ_PLIST_HIGHEST_BUFFER_NUMBER,X'80'
     L        R15,=A(STACKQUERY)
     BALR     R14,R15
```

Appendix B. Symbols Defined in Binding Files

Some CSL routines require language binding files that define symbols for values used as input parameters and for return and reason codes. These symbols are part of the API; programmers should use the symbols rather than the numeric codes. This mapping of codes to symbols is provided for interpreting error messages from parts of the programming interface that use the numeric codes.

The following tables list the symbols and their values used for input parameters and for return and reason codes. These tables are using the Stack routines as an example. See the appropriate binding files for the symbols for other routines.

Also, Ada does not support binding files and must use numeric values for parameters.

Table 70. Return Codes and Symbols for Program Stack Routines

Code	Symbol	Code	Symbol
0	vm_success	8	vm_error
4	vm_warning		

Table 71. Reason Codes and Symbols for Program Stack Routines

Code	Symbol	Code	Symbol
0	vm_stk_success	99554	vm_stk_bad_line_length
90215	vm_stk_bad_buffer_length	99555	vm_stk_bad_pad
96100	vm_stk_insufficient_storage	99556	vm_stk_bad_order
99550	vm_stk_bad_buffer	99557	vm_stk_empty
99551	vm_stk_bad_buffer_number	99558	vm_stk_no_buffer
99552	vm_stk_bad_case	99559	vm_stk_top_buffer_empty
99553	vm_stk_bad_drop_top	99560 *	vm_stk_unexpected_error
* If you receive the unexpected-error reason code, contact your system programmer.			

Table 72. Input Parameters and Symbols for Program Stack Routines

Symbol	Value	Symbol	Value
vm_stk_drop_top	1	vm_stk_pad_none	2
vm_stk_fifo	1	vm_stk_pad_nulls	1
vm_stk_lifo	0	vm_stk_query_all_buffers	-1
vm_stk_no_drop_top	0	vm_stk_query_top_buffer	-2
vm_stk_no_shift	0	vm_stk_shift	1
vm_stk_pad_blanks	0		

Appendix C. Extract/Replace Supported Information Names

To obtain or change specific subsets of system information, the Extract/Replace routine (DMSERP) requires that you provide the associated Information Name (*infoname*).

The tables in this chapter list the Information Names supported by Extract/Replace. Each table is an information set. An information name can be used as a search argument (*search_arg_name* in the DMSERP parameter list) for another information name in the same set unless there is an "N" in the search argument (Srch Arg) column. In most cases, search arguments can be used for accessing information names in an information set. The exceptions are noted in the introductions to each set.

Not all information names can be used as search arguments, since not all combinations make sense. It is also important to consider the capabilities of the language you are using when you extract information. The information returned may not be:

- In a format accessible to the language you are using (such as bit strings or addresses)
- Meaningful or useful in the language you are using (such as an address being returned to a FORTRAN program).

The tables contain this information:

Field Label

Information Name
name of information to be extracted, replaced, or searched for

Type
data type of the information:

Numeric
Four-byte binary

Indicator
A one-byte character string containing a status code ("0" or "1") for the program, the system, or a device.

CL*n*
A fixed-length character string, *n* bytes long.

Address
Four-byte, unsigned binary. An address in the user's primary address space.

Rep
tells whether the information can be replaced (Y) or only extracted (N)

Description
description of the information

Srch Arg
tells whether the information name can be used as a search argument (Y) or cannot be used as a search argument (N)

Some information names contain abbreviations:

ABBREV	Abbreviation	CONS	Console	MISC	Miscellaneous
ACT	Active	DEV	Device	MSG	Message
ADDR	Address	INT	Interrupt	NUM	Number
BLK	Block	LEN	Length	REC	Record

CLOS	Close	LIB	Library	ROU	Routine
CMD	Command	LVL	Level		

Minidisk and Directory Set

A minidisk or SFS directory must be accessed in order for information about it to be extracted. The information in this set occurs more than once in system storage.

Table 73. Minidisk and Directory Information

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CMS_READ_ONLY_DISK			
Indicator	N	The extracted value indicates whether a CMS disk or directory is accessed as read-only. A value of '1' indicates that the first accessed disk or directory matching the search criteria is read-only. A value of '0' indicates that the disk or directory is not accessed read-only.	Y
CMS_READ_WRITE_DISK			
Indicator	N	The extracted value indicates whether a CMS disk or directory is accessed as read/write. A value of '1' indicates that the first accessed disk or directory matching the search criteria is read/write. A value of '0' indicates that the disk is not accessed as read/write.	Y
READ_ONLY_EXTEND			
Indicator	N	The extracted value indicates whether a disk or directory has read-only extensions. A value of '1' indicates that the first accessed disk or directory matching the search criteria has a read-only extension. A value of '0' indicates that the disk does not have read-only extensions.	Y
ACCESS_MODE			
CL1	N	The extracted value is the mode letter (A,...,Z) for the first accessed disk or directory matching the search criteria.	Y
ACCESS_MODE_EXTEND			
CL1	N	The extracted value is the access mode of the disk or directory of which the first disk or directory matching the search criteria is an extension. If blank on return, the first disk or directory meeting the search criteria is not an extension of another mode.	Y
NUM_FILES			
Numeric	N	The extracted value is the number of user files on the disk matching the search criteria.	N
DISK_BLKSIZE			
Numeric	N	The extracted value indicates the block size of the first disk or directory matching the search argument criteria, if specified.	Y
ACT_DISK_FILE_SYSTEM			

Table 73. Minidisk and Directory Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Numeric	N	The extracted value indicates which file system the accessed disk (file mode) belongs to: **1** indicates that the active disk is a minidisk **2** indicates that the active disk is an SFS directory When this information name is used as a search argument, only the comparison types EQ and NE are valid.	Y
ACT_DISK_CONTROL_LVL			
Numeric	N	The extracted value indicates the level of disk control in effect: **1** indicates an active disk consisting of a minidisk with minidisk-level control in effect. **2** indicates an active disk consisting of an SFS filelevel control directory **3** indicates an active disk consisting of an SFS directory control directory. When this information name is used as a search argument, only the comparison types EQ and NE are valid.	Y
ACT_DISK_SPACE_TOTAL			
Numeric	N	The extracted value is the total number of blocks in the minidisk or top directory. (See note below).	Y
ACT_DISK_SPACE_USED			
Numeric	N	The extracted value is the number of blocks in the minidisk or top directory that are currently used. (See the note at the end of this table.) For a minidisk, to calculate the actual disk space used, you must add the system reserved blocks to this value. The actual disk space used = ACT_DISK_SPACE_USED + ACT_DISK_SPACE_ARES. This calculated value is equal to the BLKS USED value displayed by the QUERY DISK command, which includes the system reserved blocks in its calculation.	Y
ACT_DISK_SPACE_LEFT			

Table 73. Minidisk and Directory Information (continued)			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Numeric	N	The extracted value is the number of blocks in the minidisk or top directory that are left for future use. (See the note at the end of this table). For a minidisk, to calculate the actual disk space left, you must subtract the system reserved blocks from this value. The actual disk space left = ACT_DISK_SPACE_USED - ACT_DISK_SPACE_ARES. This calculated value is equal to the BLKS LEFT value displayed by the QUERY DISK command, which includes the system reserved blocks in its calculation.	Y
ACT_DISK_SPACE_ARES			
Numeric	N	The extracted value is the number of minidisk blocks that are reserved for directory data and pointer blocks and for allocation map data and pointer blocks. This value is useful in determining the actual number of blocks in a minidisk that are currently used or that are left for future use. See ACT_DISK_SPACE_USED and ACT_DISK_SPACE_LEFT.	Y

Note: The ACT_ information names provide information about SFS directories like that provided about minidisks, with these restrictions:

- Space information can only be extracted for top (root) directories.
- The USED and LEFT values reflect only currently committed blocks.
- Blocks used because of changes to objects in subdirectories are generally not reflected in the space values for the top directory until you release and reaccess the directory. Alternatively, you can use the DMSQLIMU CSL routine to obtain the status of your storage space, reflecting all blocks committed, including those affected by subdirectory changes.
- Administrative changes to TOTAL blocks are generally not reflected until you release and reaccess the directory. Alternatively, you can use is the DMSQLIMU CSL routine, which reflects the latest administrative changes in blocks assigned.
- If the server for the directory is not at CMS level 9 or later, space information is available only for the user's own top directory. No changes since the directory was accessed are reflected. To see the current status, release and reaccess the directory.

Device Set

The information in this set occurs more than once in system storage.

Table 74. Device Information			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
DEV_ADDR			
CL2	N	The extracted value is a 2-byte hexadecimal representation of the virtual device address, for example X'0191'.	Y
DEV_FLAGS			

Table 74. Device Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**

CL1	N	The extracted value is a bit string containing seven flags. The language you are using must have bit string handling capabilities in order for you to use this field. The flags are as follows:
		`DEVPSDOW X'80' Pseudo wait bit (from WAITD)`
		`DEVASAP X'40' HNDINT/HNDIO issued with ASAP`
		`DEVINTR X'20' Interrupt received for this entry`
		`DEVINTP X'10' Interrupt processed`
		`DEVKEEP X'08' KEEP past end-of-command`
		`DEVABNKP X'04' Keep beyond ABEND processing`
		`DEVAM31 X'01' SLIH invoked as AMODE=31`
			Y

DEV_TYPE			
CL1	N	The extracted value is a 1-byte hexadecimal code used to specify the device. DASD device codes are: **HEX Code** **Device Name** **X'08'** 2314 (CKD) **X'0E'** 3380 (CKD) **X'11'** 9336 (FBA) **X'26'** 3390 Model 1 **X'27'** 3390 Model 2 **X'24'** 3390 Model 3 **X'32'** 3390 Model 9	Y
DEV_NAME			
CL4	N	The extracted value is the name of the first symbolic device matching the search criteria. The possible values are ‘DSK ’, ‘TAP ’, ‘PRN ’, ‘RDR ’, ‘PCH ’, and ‘CON ’.	Y
DEV_INT_ADDR			
Address	N	The extracted value is the address of the user-supplied interrupt processing routine. This is the address set by the user through the HNDINT macro for a device.	N
DEV_MISC			
Numeric	N	The extracted information is device dependent.	Y

General System Set

The information names in this set are associated with single-occurrence data.

Note: These information names cannot be specified as search arguments.

Table 75. General System Information			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CMS_XA_MODE			
Indicator	N	The extracted value indicates whether or not the user's virtual machine is XA or XC. A value of '1' indicates XA or XC. A value of '0' indicates neither XA nor XC.	N/A
CMS_XC_MODE			
Indicator	N	The extracted value indicates whether the user is running in an XC virtual machine. A value of '1' indicates XC. A value of '0' indicates not XC.	N/A
DOS_MODE			
Indicator	N	The extracted value indicates whether the user is running in DOS mode. A value of '1' indicates that the user is in DOS mode. A value of '0' indicates that the user is not in DOS mode.	N/A
DOS_SVC_SIMULATION			
Indicator	N	The extracted value indicates whether the user is running in DOS SVC simulation mode. A value of '1' indicates that the user is. A value of '0' indicates that the user is not.	N/A
DOS_VSAM_ON			
Indicator	N	The extracted value indicates whether the user is running in DOS VSAM mode. A value of '1' indicates that the user is. A value of '0' indicates that the user is not.	N/A
NO_TYPE_EXEC			
Indicator	Y	The extracted value indicates whether typing has been halted by an exec. A value of '1' means an exec was used to halt typing. A value of '0' means typing has not been halted by an exec.	N/A
NO_TYPE_HT			
Indicator	Y	The extracted value indicates whether typing has been halted by a Halt Type (HT) command. A value of '1' means it has. A value of '0' means an HT command is not in effect.	N/A
NO_TIME			
Indicator	Y	The extracted value indicates whether the time of day will appear in the CMS ready message. A value of '1' means the time will **not** be displayed. A value of '0' means the time **will** appear with the ready message.	N/A
BATCH_MONITOR_ON			
Indicator	N	The extracted value indicates whether the batch monitor is active. A value of '1' means it is. A value of '0' means it is not.	N/A
GRAPHICS_CONSOLE			
Indicator	N	The extracted value indicates whether the user is on a graphics console. A value of '1' means yes. A value of '0' means no.	N/A
NO_IMPLIED_EXEC_CMD			

Table 75. General System Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Indicator	Y	The extracted value indicates whether searching for execs during command resolution is inhibited. When the search for execs is inhibited, exec commands must be preceded by the word 'exec' in order to be processed. A value of '1' means the search is inhibited. A value of '0' means it is not.	N/A
NO_IMPLIED_CP_CMD			
Indicator	Y	The extracted value indicates whether the search for CP commands is inhibited. When the search for CP commands is inhibited, CP commands must be preceded by '#CP' in order to be processed. A value of '1' means the search is inhibited. A value of '0' means it is not.	N/A
NO_STANDARD_SYNONYMS			
Indicator	Y	The extracted value indicates whether the standard synonyms table is to be searched. When the table is not searched, commands must be fully spelled out in order to be recognized and processed. A value of '1' means the table is **not** to be searched. A value of '0' means it is searched.	N/A
NO_CMD_ABBREV			
Indicator	Y	The extracted value indicates whether the command abbreviations table is to be searched. When the table is not searched, commands must be fully spelled out in order to be recognized and processed. A value of '1' means the table is **not** to be searched. A value of '0' means it is searched.	N/A
NO_AUTO_PAGE_RELEASE			
Indicator	Y	The extracted value indicates whether the automatic page release is inhibited. A value of '1' means that it is inhibited.	N/A
NO_AUTO_CONS_READ			
Indicator	Y	The extracted value indicates whether the automatic VM console read is inhibited. A value of '1' means that it is inhibited.	N/A
COMPILER_SWITCH			
Indicator	Y	The extracted value indicates the compiler switch setting. A value of '1' means the switch is 'on'. CMS clean-up procedures turn the COMPILER_SWITCH setting off before leaving an application program. So while Extract/Replace may be used to turn this setting on, the 'on' setting is only maintained within the application program. The setting is turned off (the default) upon exiting the application program.	N/A
ADDR_SAVE_AREA			
Address	N	The extracted value is the address of the current save area. The actual address of the save area changes many times dynamically while the application program is running. As part of the CMS clean-up procedure, it is reset to zero upon exiting the application program.	N/A
MODULE_START_ADDR			

Table 75. General System Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Address	N	The extracted value is the starting address of a module that has been loaded using the LOAD command.	N/A
CMD_LIST			
CL536	N	The extracted value is the command list. The last command entered is the first information found when using CMD_LIST. If the command list is extracted into a buffer and then displayed, the user must be aware of the limitations on displaying fields when defining the buffer.	N/A
NUM_FINISHED_READS			
Numeric	N	The extracted value is the sum of entries on the program stack and in the input buffer waiting to be read by the console.	N/A
NUM_PENDING_WRITES			
Numeric	N	The extracted value is the number of lines on the console output buffer waiting to be written to the terminal.	N/A
CMS_LEVEL_MSG			
CL*n*	N	The extracted value is the message returned by the CMS QUERY CMSLEVEL command. Information about the functional and service levels of CMS can also be obtained with the DMSQEFL CSL routine. The length is obtained from CMS_LEVEL_MSG_LEN.	N/A
CMS_LEVEL_MSG_LEN			
Numeric	N	The extracted value is the length of the CMS level message. This information is useful in determining the size of buffer needed to extract the CMS level message itself.	N/A
LOADLIB_NAMES			
CL*n*	N	The extracted value is a character string of 8-byte names of all of the loadlibs that were specified on the most recent GLOBAL LOADLIB command. The names may be padded on the right with blanks to obtain the proper length. The length of this character string can be obtained from LOADLIB_LEN.	N/A
LOADLIB_LEN			
Numeric	N	The extracted value is the length of the character string returned if LOADLIB_NAMES is extracted. This information is useful in determining the size of buffer needed to extract LOADLIB_NAMES.	N/A
LOADLIB_NUM			
Numeric	N	The extracted value is the number of LOADLIBs specified on the most recent global LOADLIB command.	N/A
TXTLIB_NAMES			
CL*n*	N	The extracted value is a character string of 8-byte names of all of the txtlibs that were specified on the most recent GLOBAL TXTLIB command. The names may be padded on the right with blanks. The length of this character string can be obtained from TXTLIB_LEN.	N/A

Table 75. General System Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
TXTLIB_LEN			
Numeric	N	The extracted value is the length of the character string returned if TXTLIB_NAMES is extracted. This information is useful in determining the size of buffer needed to extract TXTLIB_NAMES.	N/A
TXTLIB_NUM			
Numeric	N	The extracted value is the number of txtlibs specified on the most recent global TXTLIB command.	N/A
MACLIB_NAMES			
CL*n*	N	The extracted value is a character string of 8-byte names of all of the MACLIBs that were specified on the most recent global MACLIB command. The names may be padded on the right with blanks. The length of this character string can be obtained from MACLIB_LEN.	N/A
MACLIB_LEN			
Numeric	N	The extracted value is the length of the character string returned if MACLIB_NAMES is extracted. This information is useful in determining the size of buffer needed to extract MACLIB_NAMES.	N/A
MACLIB_NUM			
Numeric	N	The extracted value is the number of maclibs specified on the most recent global MACLIB command.	N/A
VIRTUAL_MEMORY_SIZE			
Numeric	N	The extracted value is the number of bytes of storage (in hexadecimal) in the user's virtual machine.	N/A
NUM_TERM_INPUT_LINES			
Numeric	N	The extracted value indicates the number of logical lines in the terminal input buffer. Note that this value provides the same information as the REXX EXTERNALS function.	N/A
OSRUN_ACTIVE			
Indicator	N	OSRUN_ACTIVE has a value of ‘0’ when no active application has been invoked with the CMS OSRUN command. When an application is invoked with OSRUN, OSRUN_ACTIVE is set to ‘1’ and is not reset to ‘0’ until the application finishes. ‘1’ is returned to all routines called while the application is active, including those called using CMS conventions.	N/A
YEAR2000_SUPPORT			
Indicator	N	The extracted value indicates if the users machine is enabled for Year 2000 support. A value of '1' indicates the machine is enabled to process 4-digit year input and output from commands and applications. A value of '0' indicates the machine is restricted to processing only 2-digit year input and output. **Note:** For this indicator to be '1', both CP and CMS must be enabled.	N/A

OS Simulation Set

This set contains information about devices and files associated with data definitions created using the FILEDEF command.

Table 76. OS Simulation Information

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CRT_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a graphics display. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a graphics display. A value of '0' means the device is not a graphics display.	Y
PUNCH_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a punch device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a punch device. A value of '0' means the device is not a punch device.	Y
DISK_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a disk device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a disk device. A value of '0' means the device is not a disk device.	Y
TAPE_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a tape device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a tape device. A value of '0' means the device is not a tape device.	Y
CONS_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a console terminal device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a console terminal device. A value of '0' means the device is not a console terminal.	Y
READER_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a reader device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a reader device. A value of '0' means the device is not a reader device.	Y
PRINTER_DEV			

Table 76. OS Simulation Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a printer device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a printer device. A value of '0' means the device is not a printer device.	Y
DUMMY_DEV			
Indicator	N	The extracted value indicates whether the device associated with the first data definition matching the search criteria is a dummy device. Data definitions are created by using the FILEDEF command. A value of '1' means the first device matching the search criteria is a dummy device. A value of '0' means the device is not a dummy device.	Y
DATA_SET_DEV_TYPE			
CL4	N	The extracted value indicates the specific type of device associated with the first data definition matching the search criteria. This gives a quick reference to the type of device. The possible values are 'CRT ', 'DSK ', 'TAP ', 'RDR ', 'CON ', 'PCH ', 'PRT ', and 'DUM '.	N
DATA_SET_MODE			
CL2	N	The extracted value indicates the file mode associated with the first data definition matching the search criteria. File modes are only associated with FILEDEF commands specifying the DISK device option.	Y
DATA_SET_NAME			
CL8	N	The extracted value is the name of the file associated with the first data definition matching the search criteria. File names are only associated with FILEDEF commands specifying the DISK device option.	Y
DATA_SET_TYPE			
CL8	N	The extracted value is the type of the file associated with the first data definition matching the search criteria. File names are only associated with FILEDEF commands specifying the DISK device option.	Y
DATA_DEFINITION_NAME			
CL8	N	The extracted value is the data definition name associated with the first data definition matching the search criteria. Data definitions are created by using the FILEDEF command.	Y
DATA_SET_REC_FORMAT			

Table 76. OS Simulation Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CL1	N	The extracted value is 1 byte of flag information. The possible combinations of this byte are: B'10000000' - F B'10010000' - FB B'01000000' - V B'01010000' - VB B'10001000' - FS B'01001000' - VS B'10011000' - FBS B'01011000' - VBS B'11000000' - U B'10000100' - FA B'01000100' - VA B'10010100' - FBA B'01010100' - VBA B'10011100' - FBSA B'01011100' - VBSA B'10001100' - FSA B'01001100' - VSA B'11000100' - UA B'10000010' - FM B'01000010' - VM B'10010010' - FBM B'01010010' - VBM B'10011010' - FBSM B'01011010' - VBSM B'10001010' - FSM B'01001010' - VSM B'11000010' - UM Where: **F** fixed **V** variable **B** blocked **S** standard/spanned **A** ASA **M** machine **U** undefined See the following 6 information names for more information.	N
DATA_SET_FIXED			

Table 76. OS Simulation Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Indicator	N	The extracted value indicates whether the record format associated with the first data definition matching the search criteria is fixed. A value of ‘1’ indicates fixed record format.	Y
DATA_SET_VARIABLE			
Indicator	N	The extracted value indicates whether the record format associated with the first data definition matching the search criteria is variable. A value of ‘1’ indicates variable record format.	Y
DATA_SET_BLOCKED			
Indicator	N	The extracted value indicates whether the record format associated with the first data definition matching the search criteria is blocked. A value of ‘1’ indicates blocked record format.	Y
DATA_SET_STAND_SPAN			
Indicator	N	The extracted value indicates whether the record format associated with the first data definition matching the search criteria is fixed standard (if DATA_SET_FIXED is also on) or variable spanned (if DATA_SET_VARIABLE is also on). A value of ‘1’ indicates fixed standard or variable spanned.	Y
DATA_SET_ASA			
Indicator	N	The extracted value indicates whether ASA print control characters are associated with the first data definition matching the search criteria. A value of ‘1’ indicates that there are.	Y
DATA_SET_MACHINE			
Indicator	N	The extracted value indicates whether machine print control characters are associated with the first data definition matching the search criteria. A value of ‘1’ indicates that there are.	Y
DATA_SET_REC_LEN			
Numeric	N	The extracted value is the logical record length associated with the first data definition matching the search criteria.	Y
SPECIAL_PROCESS_ROU			
Address	Y	The extracted value is the address of a user-supplied special processing routine.	N
SPECIAL_PROCESS_CLOS			
Indicator	Y	The extracted value indicates whether the user-supplied special processing routine (SPECIAL_PROCESS_ROU) is to be used during close. A value of ‘1’ means it will be used. A value of ‘0’ means no user-supplied special processing routine is to be used.	Y
SPECIAL_PROCESS_OPEN			
Indicator	Y	The extracted value indicates whether the user-supplied special processing routing (SPECIAL_PROCESS_ROU) is to be used during open. A value of '1' means it will be used. A value of '0' means no user-supplied special processing routine is to be used.	Y

Table 76. OS Simulation Information (continued)			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
LOWER_CASE_CONS_IO			
Indicator	N	The extracted value indicates whether all terminal input data is to be retained just as it is typed in. A value of '1' means it is.	Y

File Set

To extract information about a file, the minidisk or SFS directory on which the file resides must be accessed. The information in this set occurs more than once in system storage.

Table 77. File Information			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
FILE_NAME			
CL8	N	The extracted value is the file name of the first CMS file matching the search criteria.	Y
FILE_TYPE			
CL8	N	The extracted value is the file type of the first CMS file matching the search criteria.	Y
FILE_MODE			
CL1	N	The extracted value is the file mode (letter only) of the first CMS file matching the search criteria.	Y
FILE_MODE_NUM			
CL1	N	The extracted value is the file mode number of the first CMS file matching the search criteria.	Y
FILE_REC_FORMAT			
CL1	N	The extracted value is the record format of the first CMS file matching the search criteria. Possible values are: **'F'** fixed-length records **'V'** variable-length records Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object.	Y
FILE_REC_LEN			
Numeric	N	The extracted value is the logical record length of the first CMS file matching the search criteria. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object.	Y
FILE_ORIGIN_POINTER			

Table 77. File Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Numeric	N	For minidisk files, the extracted value is the block number for the top block of the file. This block number is unique to the minidisk containing the file. For SFS files, the extracted value is the logical block number for the top block of the file. This number is in the range of 1 to the number of physical blocks in the file. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object.	Y
FILE_BLOCK_COUNT			
Numeric	N	The extracted value is the number of data blocks of disk space physically or logically occupied by the first CMS file matching the search criteria. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object.	Y
FILE_NUM_REC			
Numeric	N	The extracted value is the number of records in the first CMS file matching the search criteria. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object.	Y
FILE_DATE_CENTURY			
Indicator	N	The extracted value is determined by the time frame that the first file matching the search criteria was created or last updated. A value of '0' indicates that the YY from FILE_DATE_TIME or FILE_DATE_TIME_C is within the time frame 01/01/1900 and 12/31/1999. A value of '1' indicates that the YY lies within the time frame 01/01/2000 and 12/31/2099. This applies to only one file.	Y
FILE_DATE_TIME			
CL6	N	The extracted value is determined by the date and time that the first file matching the search criteria was created or last updated. The date and time are coded in 12 half-bytes: YYMMDDHHMMSS. Each half-byte is interpreted as a hexadecimal character. For example, 10:45:14 a.m. on March 4, 1988, is rendered as the 6-byte hexadecimal string 880304104514. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object. Use the FILE_DATE_CENTURY indicator to determine the 4-digit year.	Y
FILE_DATE_TIME_C			
CL12	N	The extracted value is date and time that the first file matching the search criteria was created or last updated. This date/time is in the form: YYMMDDHHMMSS. Each byte of the date/time string is interpreted as a character. For example, 10:45:15 am on March 4, 1988, is rendered as the 12-byte character string 880304104515. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object. Use the FILE_DATE_CENTURY indicator to determine the 4-digit year.	N
FILE_DIRECTORY_ID			

Table 77. File Information (continued)			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CL144	N	The extracted value is the fully-qualified name of the directory in which the first SFS file matching the search criteria resides. A fully-qualified directory ID consists of an 8-byte file pool ID immediately followed by an 8-byte owner ID immediately followed by up to eight 16-byte qualifiers. Each field must be padded with blanks if necessary. If the file is a minidisk file, return code 133 is returned indicating that there is no directory ID associated with the file.	Y
FILE_ESM_PROTECT			
Indicator	N	The extracted value indicates whether the first SFS file matching the search criteria is protected by an external security manager (ESM). A value of '1' indicates that there is ESM protection. A value of '0' indicates there is no ESM protection. Return code 133 indicates that the file matching the search criteria is either a minidisk file, or a revoked or erased alias, or an external object.	Y
FILE_SYSTEM			
Numeric	N	The extracted value indicates which file system the file belongs to. **1** indicates that it is a minidisk file **2** indicates that it is an SFS file.	N
FILE_CONTROL_LVL			
Numeric	N	The extracted value indicates the level of control in effect for the file: **1** indicates a minidisk file with minidisk-level control **2** indicates an SFS file with file-level control. **3** indicates an SFS file with directory-level control. Return code 133 indicates that the file matching the search criteria is a revoked or erased alias, or an external object.	N

Active File Set

This set contains information names data about open files. The information in this set occurs more than once in system storage.

Table 78. Active File Information			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
ACT_FILE_NAME			
CL8	N	The extracted value is the file name of the first active CMS file matching the search criteria.	Y

Table 78. Active File Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
ACT_FILE_TYPE			
CL8	N	The extracted value is the file type of the first active CMS file matching the search criteria.	Y
ACT_FILE_MODE			
CL1	N	The extracted value is the file mode letter of the first active CMS minidisk file (or SFS file opened using the FS macros) matching the search criteria. If the file matching the search criteria is an SFS file that was opened by DMSOPEN or DMSOPDBK using a directory name rather than a file mode, return code 133 is returned indicating that no file mode is associated with this file.	Y
ACT_FILE_TOKEN			
CL8	N	The extracted value is the file token (the one passed back to the user from DMSOPEN, DMSOPDBK, or DMSOPBLK) of the first active CMS file matching the search criteria. If DMSOPEN, DMSOPDBK, or DMSOPBLK was not used to open the file, return code 133 is returned indicating that no token is associated with the file.	Y
ACT_FILE_REC_FORMAT			
CL1	N	The extracted value is the record format of the first active CMS file matching the search criteria.	Y
ACT_FILE_REC_LEN			
Numeric	N	The extracted value is the logical record length of the first active CMS file containing at least one record and matching the search criteria. If the file does not contain at least one record or was not opened using the DMSOPEN routine with the *lrecl* parameter specified, return code 133 is returned indicating that a record length has not yet been established.	Y
ACT_FILE_NUM_REC			
Numeric	N	The extracted value is the number of records in the first active CMS file matching the search criteria.	Y
ACT_FILE_DATE_CENTRY **Note:** Information names are restricted to a length of 20 characters. Therefore, the spelling of "_CENTRY" is intentional.			
Indicator	N	The extracted value is determined by the time frame that the first active file matching the search criteria was created or last updated. A value of '0' indicates that the YY from ACT_FILE_DATE_TIME or ACT_FILE_DATE_TIME_C is within the time frame 01/01/1900 and 12/31/1999. A value of '1' indicates that the YY lies within the time frame 01/01/2000 and 12/31/2099. This applies to only one file.	Y
ACT_FILE_DATE_TIME			

Table 78. Active File Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CL6	N	The extracted value is the character format date and time that the first active file matching the search criteria was created or last updated. This date/time is in the form: YYMMDDHHMMSS. Each half of each byte is interpreted as a hexadecimal character. For example, 10:45:15 am on March 4, 1988, is rendered as the 6-byte hexadecimal string 880304104515. A return code of 133 is returned if a date and time have not yet been established for a file. Use the ACT_FILE_DATE_CENTRY indicator to determine the 4-digit year.	Y
ACT_FILE_DATE_TIME_C			
CL12	N	The extracted value is the date and time that the first active file matching the search criteria was created or last updated. This date/time is in the form: YYMMDDHHMMSS. Each byte of the date/time string is displayed as the character representation of the hexadecimal value. For example, 10:45:15 am on March 4, 1988, is rendered as the 12-byte character string 880304104515. A return code of 133 is returned if a date and time have not yet been established for a file. Use the ACT_FILE_DATE_CENTRY indicator to determine the 4-digit year.	N
OPEN_INTENT			
Numeric	N	The open intent of the active file is returned: **1** indicates NEW intent **2** indicates READ intent **3** indicates WRITE intent **4** indicates REPLACE intent.	N
READ_RECORD_NUMBER			
Numeric	N	The record number of the next record to be read is returned. Return code 133 is returned for files opened by DMSOPBLK indicating that access to individual records of a file is not supported.	N
WRITE_RECORD_NUMBER			
Numeric	N	The record number of the next record to be written is returned. If a file was opened by DMSOPBLK or DMSOPDBK, or was explicitly opened for READ by FSOPEN or DMSOPEN, return code 133 is returned to indicate that WRITE access to records of this file has not been established.	N
OPEN_ROUTINE			

Table 78. Active File Information (continued)			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
Numeric	N	The extracted value indicates which open routine was used to open the active SFS file: **1** indicates that the FS macro interface was used. **2** indicates that DMSOPEN was used **3** indicates that DMSOPBLK was used. **4** indicates that DMSOPDBK was used.	N
ACT_FILE_SYSTEM			
Numeric	N	The extracted value indicates which file system the open file belongs to: **1** indicates an open minidisk file **2** indicates an open SFS file **3** indicates an open BFS file.	N

Shared File Set

The information names in this set are associated with single-occurrence data.

Note: These information names cannot be specified as search arguments.

Table 79. Shared File Information			
Information Name			
Type	**Rep**	**Description**	**Srch Arg**
INITIAL_FILE_POOL_ID			
CL8	N	The extracted value is the primary file pool ID that is set at IPL time.	N/A
CURRENT_FILE_POOL_ID			
CL8	N	The extracted value is the file pool ID set by the most recent SET FILEPOOL command.	N/A
DEFAULT_WORK_UNIT_ID			
Numeric	N	The extracted value is the current default work unit ID.	N/A
DEFAULT_RECOVERY			

Table 79. Shared File Information (continued)

Information Name			
Type	**Rep**	**Description**	**Srch Arg**
CL7	N	The extracted value is a 7-byte character string with each byte containing an indicator representing the default value to be assigned to the recoverability attribute associated with a file mode number in SFS. The bytes represent file mode numbers in ascending order (‘0’... ‘6’). Possible values are: **1** RECOVER - Updates to the file will be backed out as a result of an application initiated rollback request. **0** NORECOVER - Updates to the file will not be backed out as a result of an application initiated rollback request. Each SFS file is associated with a file mode number. If the recoverability characteristics of an SFS file are not specified at the time of creation, the default recoverability attributes associated with the file mode number take effect. The system default recoverability attributes normally indicate RECOVER. They can be changed for the duration of an application (for example, until Command-End) by using the DMSPUSHA and DMSPOPA CSL routines. You can also specify recoverability of an SFS file using the DMSOPEN CSL routine.	N/A
DEFAULT_OVERWRITE			
CL7	N	The extracted value is a 7-byte character string with each byte containing an indicator representing the default value to be assigned to the overwrite attribute associated with a file mode number in SFS. The bytes are ordered to represent file mode numbers in ascending order (0 - 6). The possible values of each byte are as follows: **1** INPLACE - Updates to the existing file data blocks will be done in place. **0** NOTINPLACE - All updates to the file will be shadowed for read. The overwrite characteristics for an SFS file may be explicitly specified by using special keywords when invoking the DMSOPEN Callable Services Library (CSL) routine. Each SFS file is associated with a file mode number. If the overwrite characteristics of an SFS file are not specified at the time of creation, the default overwrite attributes associated with the file mode number take effect. The system default overwrite attributes normally indicate NOTINPLACE. They may be changed for the duration of an application (for example, until Command-End) by using the DMSPUSHA and DMSPOPA CSL routines.	N/A

Appendix D. Return Codes

When a VMLIB CSL routine is called, a return code is issued by either the called routine or the calling interface. If the return code from the called routine is zero, the reason code returned from the routine is also zero. If the return code from the called routine is nonzero, the reason code provides additional information about the cause. (For information about reason codes, see Appendix E, “Reason Codes,” on page 601.)

Return Codes Issued by VMLIB Routines

Most VMLIB CSL routines issue the following return codes:

Code
Meaning

0
The operation was successful.

4
The operation was successful, but a warning condition was encountered.

8
The operation was unsuccessful.

12
The operation was unsuccessful, and the current work unit was rolled back.

Note: When the return code is 8, the work unit may still be considered active even though the request failed. Before an application can issue an atomic program function on the same work unit for the same file pool, it must first issue a commit or rollback request for that work unit.

The Commit routine (DMSCOMM) and all VMLIB CSL routines with the COMMIT option can also issue the following return codes:

Code
Meaning

16
The work was committed, but the state may not be consistent. See the associated reason code.

20
The work was rolled back, but the state may not be consistent. See the associated reason code.

The Rollback routine (DMSROLLB) can also issue these return codes:

Code
Meaning

12
Rollback was successful; however, the rollback was caused by an event such as a failure of one of the protected resources.

20
Rollback was successful, but one or more protected resources may have committed changes.

The user-coded File Pool Storage Use Exit routine (DMSSFSEX) can also issue this return code:

Code
Meaning

5
The requested function is not supported by the exit called. Further calls to the exit for this file pool function are suppressed. The file pool server takes its default action.

Some VMLIB CSL routines may not issue the return codes described in this section but instead issue their own special return codes. Those special return codes are included in the description of each routine.

Return Codes Issued by the Calling Interface

The following return codes are generated when the interface used to call a CSL routine (direct call, DMSCSL, CSLFPI macro, REXX CSL function, or REXX ADDRESS OPENVM instruction) encounters a problem, such as parameters not matching what is in the template file. These codes are returned in the *retcode* parameter. (For the ADDRESS OPENVM interface, the codes are returned in the REXX *RC* variable.)

Code
Meaning

-07
Routine was not loaded.

-08
Routine has been dropped.

-09
Insufficient virtual storage available.

-10
Too many parameters specified.

-11
Not enough parameters specified.

-12
CSL does not exist on the release.

-13
(Not issued for CSL calls from REXX.)

Call Type
Meaning

DMSCSL
Invalid parameter list format (returned in register 15 only)

Direct
Invalid Call Routing Code Segment used. The segment has incorrectly specified the multiprocessing capability of *rtnname*. The capability of the current routine version loaded is not what was specified by the Call Routing Code Segment. The call cannot be completed.

CSLFPI
CSLFPI fast path area cannot provide parameters in the standard plist format required by the currently loaded routine version.

-20
Call is not valid. (Issued only for CSL calls from REXX.)

-22
REXX argument is not valid. (Issued only for CSL calls from REXX.)

-23
Subpool create failure. (Issued only for CSL calls from REXX.)

-24
REXX fetch failure. (Issued only for CSL calls from REXX.)

-25
REXX set failure. (Issued only for CSL calls from REXX.)

-26*nnn*
Incorrect data length for parameter number *nnn*. (Issued only for CSL calls from REXX.)

-27*nnn*
Incorrect data or data type for parameter number *nnn*. (Issued only for CSL calls from REXX.)

-28*nnn*
Incorrect variable name for parameter number *nnn*. (Issued only for CSL calls from REXX.)

-29*nnn*
Incorrect length value (for example, a negative value) specified for length parameter, parameter number *nnn*. (Issued only for CSL calls from REXX.)

For more information about return codes -26*nnn* through -29*nnn*, see the *z/VM: REXX/VM Reference*.

Appendix E. Reason Codes

When the return code from a called VMLIB CSL routine is nonzero, the reason code provides additional information about the cause. (When the return code from the called routine is zero, the reason code is also zero.) Most VMLIB routines return reason codes that are specific to the routine. Those reason codes are listed in the description of the routine. The reason codes are returned as decimal numbers.

This appendix contains two sections:

- "Common Reason Codes" on page 601 is a list of additional reason codes that can be returned by most VMLIB routines.
- "All Reason Codes" on page 606 is a complete numerical list of all the reason codes returned by VMLIB routines.

The severity of the reason code can be:

Severity
Meaning

WARNING
The request was processed, or the desired state already exists. Warnings cause a return code of 4.

ERROR
The request failed. Errors cause a return code of 8 or 12.

Reason codes identified as *system error* in the description are returned for conditions that are beyond the control of the user and that probably require diagnosis by support personnel.

The term *file pool object* is used frequently in the reason code explanations. File pool objects are: files, directories, aliases, external objects, catalogs, file spaces, and storage groups. Depending on the program function and the error condition, *file pool object* can refer to any of these objects.

Common Reason Codes

The following reason codes can be returned by most VMLIB CSL routines.

Table 80. Common Record File System CSL Reason Codes

Severity	Reason Code	Description
ERROR	02000	Your attempt to use an object in a file pool conflicted with an explicit lock held on that object or another object. It could mean that you tried to open a file for writing in a directory control directory that someone already has accessed R/W.
ERROR	02100	You attempted to acquire a lock that resulted in a wait. The lock request failed because of a deadlock condition.
ERROR	02102	File pool catalog has encountered a deadlock.
ERROR	02200	You attempted to acquire a lock and the lock request failed because the requested lock would conflict with an implicit lock held by another user. Usually, this means that you either attempted to open a file for writing that was already open for writing by another user, or attempted to open a file for writing that you had opened for writing on another work unit.

Table 80. Common Record File System CSL Reason Codes (continued)

Severity	Reason Code	Description
ERROR	02300	The file pool server has rejected your request to lock a resource that is locked by a partner in a CRR protected conversation. Letting you wait for the lock would have created a deadlock. You cannot obtain the lock until either the other lock has been removed or the CRR conversation has been ended.
ERROR	30000	You do not have the required authority: • In an SFS file space, you must have file pool administration authority to perform this work unit *on another user's behalf*. The DMSGETWU routine was used to associate the user ID with the work unit. • In a BFS file space, you must have file pool administration authority to perform this task.
ERROR	30100	You do not have authority to use the specified file pool. You are not enrolled in the file pool, and PUBLIC is not enrolled.
ERROR	30300	You have attempted a remote connection to a file pool server that was started for local use only.
ERROR	30400	You have attempted a remote connection to a file pool server that was started for SSI use only.
ERROR	40000	An internal SFS limit has been exceeded.
ERROR	51010	No space for data left in catalog space.
ERROR	51020	No index space left in catalog space.
ERROR	55000	Insufficient virtual storage in the file pool server machine to process your request.
ERROR	56000	Request denied because you have a file pool catalog open for WRITE for the specified work unit ID. The only requests that will be accepted for that work unit ID are Write Catalog, Close Catalog, and Rollback.
WARNING	61621	An unresolved alias cannot be resolved.
ERROR	64500	Operation failed due to a mismatch between the level of code of your virtual machine and that of the file pool server machine.
ERROR	64600	Internal DFSMS/VM processing error. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the DFSMS/VM error code.
ERROR	64700	File or directory creation or file recall was rejected by a DFSMS/VM ACS routine. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the ACS routine return code.
ERROR	64800	Unexpected error in a DFSMS/VM ACS routine. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the ACS routine return code.

Table 80. Common Record File System CSL Reason Codes (continued)

Severity	Reason Code	Description
ERROR	64900	Communication error in DFSMS/VM. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the APPC/VM return code.
ERROR	65300	DFSMS/VM encountered an SFS error while recalling a file. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the SFS reason code.
ERROR	65400	This operation cannot be performed on the specified type of object (external object, unresolved alias, or BFS object). Or the operation is allowed for BFS objects, but the file ID passed to the routine does not identify a BFS regular file.
ERROR	66200	User selected for rollback by the SFS File Space Usage exit. The work unit will be rolled back.
ERROR	69000	Only one BFS object in a file pool may be updated on a single work unit. The work unit may not include updates to other BFS objects in the same file pool or any SFS objects in that file pool.
ERROR	71000	System error in file pool server reading a file pool catalog.
ERROR	71100	System error in file pool server writing to a file pool catalog.
ERROR	71200	The server encountered an error when it attempted to access a file.
ERROR	71300	System error in file pool server locking function.
ERROR	71400	System error in file pool server query function.
ERROR	71500	System error in file pool server storage management function.
ERROR	71600	System error in file pool server system services function.
ERROR	71700	System error in file pool server query function.
ERROR	71800	Request denied because there is a file pool server conflict with Coordinated Resource Recovery (CRR) resynchronization activity. FPERROR field FPEAUGMT contains the Recovery Token of the resynchronization activity.
ERROR	72900	The requested file pool operation is not supported for the specified file pool.
ERROR	73000	System error. Invalid request input to file pool server.
ERROR	73100	System error. Conflicting count of connected communication paths between user machine and server machine.
ERROR	73300	System error in the file pool server's data access component. Only one user is affected. To restore service, either: • Re-IPL CMS and reconnect to the server • Or use the DMSPURWU CSL routine to purge the work unit IDs.
ERROR	74000	System error. Inconsistent file pool catalogs.
ERROR	74100	System error in migrating or recalling a file.

Table 80. Common Record File System CSL Reason Codes (continued)

Severity	Reason Code	Description
ERROR	75000	Service level of your machine is not compatible with the service level of the file pool server machine.
ERROR	76002	I/O error encountered while reading the file pool catalog.
ERROR	76004	A file pool system limit has been reached.
ERROR	76050	An implicit rollback occurred. This code appears only as part of the work unit error information.
ERROR	76060	An atomic SFS request has been rolled back. This code appears only as part of the work unit error information.
ERROR	76400	Exchange log names failed with the CRR recovery server.
ERROR	78100	System error determining the node ID of the server.
ERROR	90128	Unable to obtain space on the system stack for a file system module's dynamic storage.
ERROR	90130	Insufficient free virtual storage available for file system control blocks.
ERROR	90223	Asynchronous calls may not be made from REXX.
ERROR	90545	System error. The work unit ID could not be associated with a user ID.
ERROR	90700	The input file ID resolves to a file mode letter and it is not accessed.
ERROR	94000	System error in module DMS2RQ.
ERROR	95100	System error. Invalid request type passed to SFS.
ERROR	95200	A fatal communication error occurred during a previous request. Any attempt to communicate with a file pool server will be rejected until your virtual machine is re-IPLed.
ERROR	95300	System error. An invalid operation was requested internally by SFS.
ERROR	96100	Insufficient virtual storage for a get storage request from a user's virtual machine.
ERROR	96200	System error in storage management while trying to acquire virtual storage in a user's virtual machine.
ERROR	96400	System error. Error in APPC/VM IDENTIFY function.
ERROR	96600	Error from CSL when attempting to call the SFS user accounting exit.
ERROR	96610	A CSL routine required by this routine was dropped or not loaded.
ERROR	96620	System error: This routine called another CSL routine with an incorrect number of parameters.
ERROR	96700	Request issued while a commit is in process for the specified work unit ID.
ERROR	97100	Nonzero return code received from SFS user accounting exit.

Table 80. Common Record File System CSL Reason Codes (continued)

Severity	Reason Code	Description
ERROR	97200	System error detected by APPC/VM.
ERROR	97250	You have attempted to establish more APPC/VM connections than the maximum allowed for your virtual machine, as determined by the MAXCONN value in your CP directory.
ERROR	97280	Your attempt exceeds the number of APPC/VM connections allowed for the file pool.
WARNING	97290	You have attempted to establish more APPC/VM connections than the maximum allowed for your virtual machine, as determined by the MAXCONN value in your CP directory. An inactive path has been severed and another attempt will be made to establish an APPC/VM connection for your request.
ERROR	97300	An attempt was made to use several file pool identifiers that resolve to the same file pool ID.
ERROR	97400	Sever condition returned from APPC/VM communication request. If your application receives this reason code intermittently, but the file pool is still available and other commands work correctly, the file pool server may be improperly configured. Notify your file pool administrator of this condition. (Ask the administrator to check the USERS start-up parameter value.)
ERROR	97480	Communication path severed due to an error detected by a file pool server. This code will appear only as part of the work unit error information that is provided when you specify the *wuerror* parameter.
ERROR	97500	Specified file pool is unavailable (no resource identified to APPC/VM for the specified file pool ID).
ERROR	97600	A function other than DMSCHECK was invoked while an outstanding asynchronous request was in process for the specified file pool ID and work unit ID.
ERROR	97700	A DMSCHECK request was issued when no asynchronous request was in process for the specified file pool ID and work unit ID.
ERROR	97800	For a DMSCHECK request passed to SFS, the request type did not match the request type of the outstanding asynchronous request.
ERROR	98000	System error. Unexpected return code or reason code from synchronization point manager routine DMSREG.
ERROR	98100	System error. Unexpected return code or reason code from synchronization point manager routine DMSCHREG.
ERROR	98200	System error. Unexpected return code or reason code from synchronization point manager routine DMSGETRS.
ERROR	98300	System error. Unexpected return code or reason code from synchronization point manager routine.
ERROR	98400	System error. COMMIT was specified. File attributes were not updated for a file that was written to.

Table 80. Common Record File System CSL Reason Codes (continued)		
Severity	**Reason Code**	**Description**
ERROR	98500	System error. An attempt to call a system CSL routine resulted in an unexpected return code from CSL.
WARNING	98550	System error. SFS returned an unexpected warning.
ERROR	98600	An attempt to write to a file pool was rejected because only one write-mode resource is allowed for this unit of work, and another resource is already in write mode.
ERROR	98700	The target file pool server does not support the requested function. (The file pool server is not at the proper release or service level.)
ERROR	98800	Specified file pool server does not support connections on behalf of user IDs other than the VM ID of the connecting machine: the file pool server is at z/VM Version 1 Release 1.0 or earlier.
ERROR	98900	System error. Unexpected return code or reason code from synchronization point manager routine.

All Reason Codes

The reason codes that are generated by the VMLIB CSL routines are documented.

Each reason code entry contains the name of the issuing routine, the severity, and a brief explanation. If the reason code is returned by more than one routine, an entry for each routine is documented. The numbers in parentheses distinguish entries for different routines. If the reason code can be issued by most of the VMLIB routines, "Common" is shown instead of a routine name.

VMLIB reason codes can also be returned in various system messages. In this case, any of the meanings for a particular reason code could apply. A reason code has the same general meaning no matter which CSL routine or message returns it.

“Reason Codes 02000-57080 generated by the VMLIB CSL routines” on page 606

“Reason Codes 60000-86400 generated by the VMLIB CSL routines” on page 642

“Reason Codes 90101-90472 generated by the VMLIB CSL routines” on page 673

“Reason Codes 90476-99631 generated by the VMLIB CSL routines” on page 723

Reason Codes 02000-57080 generated by the VMLIB CSL routines

Explanation, originating module, and severity user response is documented for reason codes 02000 - 57080, which are generated by VMLIB CSL routines.

02000 (1)

Explanation:
Your attempt to use an object in a file pool conflicted with an explicit lock held on that object or another object. It could mean that you tried to open a file for writing in a directory control directory that someone already has accessed R/W.

Module:
Common

Severity:
ERROR

02000 (2)

Explanation:
You cannot disable the file space because another user has an exclusive lock on the storage group.

Module:
DMSDISFS

Severity:
ERROR

02050 (1)

Explanation:
The specified lock is not held.

Module:

DMSDELOC

Severity:
WARNING

02050 (2)

Explanation:
The specified lock is not held.

Module:
DMSENAFS

Severity:
WARNING

02050 (3)

Explanation:
The specified lock is not held.

Module:
DMSENASG

Severity:
WARNING

02070

Explanation:
There are no locks held on the object by *function* RENAME.

Module:
DMSENAFS

Severity:
WARNING

02080 (1)

Explanation:
You already hold the requested lock.

Module:
DMSCRLOC

Severity:
WARNING

02080 (2)

Explanation:
You already hold the requested lock.

Module:
DMSDISFS

Severity:
WARNING

02080 (3)

Explanation:
You already hold the requested lock.

Module:
DMSDISSG

Severity:
WARNING

02100

Explanation:
You attempted to acquire a lock that resulted in a wait. The lock request failed because of a deadlock condition.

Module:
Common

Severity:
ERROR

02102

Explanation:
File pool catalog has encountered a deadlock.

Module:
Common

Severity:
ERROR

02200

Explanation:
You attempted to acquire a lock and the lock request failed because the requested lock would conflict with an implicit lock held by another user. Usually, this means that you either attempted to open a file for writing that was already open for writing by another user, or attempted to open a file for writing that you had opened for writing on another work unit.

Module:
Common

Severity:
ERROR

02300

Explanation:
The file pool server has rejected your request to lock a resource that is locked by a partner in a CRR protected conversation. Letting you wait for the lock would have created a deadlock. You cannot obtain the lock until either the other lock has been removed or the CRR conversation has been ended.

Module:
Common

Severity:
ERROR

02400 (1)

Explanation:
You already hold a SHARE lock on the object and you have requested an UPDATE or EXCLUSIVE lock.

Module:
DMSCRLOC

Severity:
ERROR

02400 (2)

Explanation:
You have requested an EXCLUSIVE lock when you or the owner already holds a SHARE lock on the file space.

Module:
DMSDISFS

Severity:
ERROR

02400 (3)

Explanation:
You have requested an EXCLUSIVE lock when you or the owner already holds a SHARE lock on the storage group.

Module:
DMSDISSG

Severity:
ERROR

02500

Explanation:
You already hold an UPDATE lock on the object and you have requested a SHARE or EXCLUSIVE lock.

Module:
DMSCRLOC

Severity:
ERROR

02600 (1)

Explanation:
You already hold an EXCLUSIVE lock on the object and have requested a SHARE or UPDATE lock.

Module:
DMSCRLOC

Severity:
ERROR

02600 (2)

Explanation:
You have requested a SHARE lock when you or the owner already holds an EXCLUSIVE lock on the file space.

Module:
DMSDISFS

Severity:
ERROR

02600 (3)

Explanation:
You have requested a SHARE lock when you or the owner already holds an EXCLUSIVE lock on the storage group.

Module:
DMSDISSG

Severity:
ERROR

02700 (1)

Explanation:
Another user holds a SHARE lock on the object and you have requested an UPDATE or EXCLUSIVE lock.

Module:
DMSCRLOC

Severity:
ERROR

02700 (2)

Explanation:
Another user holds a SHARE lock on the file space and you have requested an EXCLUSIVE lock.

Module:
DMSDISFS

Severity:
ERROR

02700 (3)

Explanation:
Another user holds a SHARE lock on the storage group for which you have requested an EXCLUSIVE lock.

Module:
DMSDISSG

Severity:
ERROR

02800

Explanation:
Another user holds an UPDATE lock on the object.

Module:
DMSCRLOC

Severity:
ERROR

02900 (1)

Explanation:
Another user holds an EXCLUSIVE lock on the object.

Module:
DMSCRLOC

Severity:
ERROR

02900 (2)

Explanation:
Another user holds an EXCLUSIVE lock on the file space.

Module:
DMSDISFS

Severity:
ERROR

02900 (3)

Explanation:
Another user holds an EXCLUSIVE lock on the storage group.

Module:
DMSDISSG

Severity:
ERROR

03000

Explanation:
Another user has a directory or file locked.

Module:
DMSDEUSR

Severity:
ERROR

03100

Explanation:
The specified file or directory, or a file in the specified directory is explicitly locked by the requester.

Module:
DMSERASE

Severity:
ERROR

10000 (1)

Explanation:
System error. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSCLBLK

Severity:
ERROR

10000 (2)

Explanation:
System error. Attempt to write a block but the file is not open for NEW or REPLACE.

Module:
DMSCLDBK

Severity:
ERROR

10000 (3)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSCLDIR

Severity:
ERROR

10000 (4)

Explanation:
System error. Attempt to write a block but the file is not open for NEW, WRITE, or REPLACE.

Module:
DMSCLOSE

Severity:
ERROR

10000 (5)

Explanation:
System error. COMMIT was specified. SFS attempted to write a block, but the file is not open for NEW, WRITE, or REPLACE.

Module:
DMSCRALI

Severity:
ERROR

10000 (6)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSCRDIR

Severity:
ERROR

10000 (7)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSERASE

Severity:
ERROR

10000 (8)

Explanation:

System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSEXIDI

Severity:
ERROR

10000 (9)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSEXIFI

Severity:
ERROR

10000 (10)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE. COMMIT was specified.

Module:
DMSEXIST

Severity:
ERROR

10000 (11)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSFILEC

Severity:
ERROR

10000 (12)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSGRANT

Severity:
ERROR

10000 (13)

Explanation:
System error. File is not open. You tried to read a file that was opened with an intent of CREATING.

Module:
DMSRDBLK

Severity:
ERROR

10000 (14)

Explanation:
File is not open using the DMSOPDBK routine with an intent of READ.

Module:
DMSRDDBK

Severity:
ERROR

10000 (15)

Explanation:
System error. File is not open by DMSOPEN.

Module:
DMSREAD

Severity:
ERROR

10000 (16)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSRENAM

Severity:
ERROR

10000 (17)

Explanation:
System error. COMMIT was specified. Attempt to write a block but a file is not open for NEW, WRITE, or REPLACE.

Module:
DMSREVOK

Severity:
ERROR

10000 (18)

Explanation:
File is not open with intent NEW, WRITE, or REPLACE.

Module:
DMSWRBLK

Severity:
ERROR

10000 (19)

Explanation:

File is not open with intent NEW or REPLACE.

Module:
DMSWRDBK

Severity:
ERROR

10000 (20)

Explanation:
File is not open with intent NEW, WRITE, or REPLACE.

Module:
DMSWRITE

Severity:
ERROR

10050 (1)

Explanation:
No write was done for a new file created as a result of OPEN NEW or OPEN WRITE. Upon closing, the file no longer exists.

Module:
DMSCLBLK

Severity:
WARNING

10050 (2)

Explanation

No file has been created. The file was empty when you tried to close it, because:

- Either DMSOPDBK NEW was specified and the file was not written to
- Or an empty file was not created because either ALLOWEMPTY was not specified on the OPEN, or the file pool or minidisk does not support empty files (file pool server is not at the z/VM Version Release 1.1 level or higher.)

Module:
DMSCLDBK

Severity:
WARNING

10050 (3)

Explanation

No file has been created. The file was empty when you tried to CLOSE it, either because:

- OPEN NEW was specified and the file was not written to
- Or because OPEN WRITE was specified for a file that did not exist and the file was not written to. An empty file was not created either because ALLOWEMPTY was not specified on the OPEN or because the file pool or minidisk does not support empty files (file pool server is not at the z/VM Version 1 Release 1.1 level or higher).

Module:
DMSCLOSE

Severity:
WARNING

10070 (1)

Explanation:
Open intent was REPLACE, but no write was issued. Original file is kept.

Module:
DMSCLBLK

Severity:
WARNING

10070 (2)

Explanation:
The open intent was REPLACE and no write was issued. The original SFS file is kept, either because DMSOPEN was not specified with ALLOWEMPTY, or because the file pool server is not at the z/VM Version 1 Release 1.1 level or higher and does not support empty files.

Module:
DMSCLOSE

Severity:
WARNING

10070 (3)

Explanation:
The open intent was REPLACE and no write was issued. The original file is kept, either because DMSOPDBK was not specified with ALLOWEMPTY, or because the file pool server is not at the z/VM Version 1 Release 1.1 level or higher and does not support empty files.

Module:
DMSCLDBK

Severity:
WARNING

10100 (1)

Explanation:
System error. Conflicting file attributes. Number of blocks does not match MAXBLOCK or not all blocks have been written.

Module:
DMSCLBLK

Severity:
ERROR

10100 (2)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSCLDBK

Severity:
ERROR

10100 (3)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSCLDIR

Severity:
ERROR

10100 (4)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSCLOSE

Severity:
ERROR

10100 (5)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSCRALI

Severity:
ERROR

10100 (6)

Explanation:
System Error. COMMIT was specified. Conflicting file attributes.

Module:
DMSCRDIR

Severity:
ERROR

10100 (7)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSERASE

Severity:
ERROR

10100 (8)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSEXIDI

Severity:
ERROR

10100 (9)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSEXIFI

Severity:
ERROR

10100 (10)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSEXIST

Severity:
ERROR

10100 (11)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSFILEC

Severity:
ERROR

10100 (12)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSGRANT

Severity:
ERROR

10100 (13)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSRENAM

Severity:
ERROR

10100 (14)

Explanation:
System error. COMMIT was specified. Conflicting file attributes.

Module:
DMSREVOK

Severity:
ERROR

10210

Explanation:
Error obtaining UID or GID

Module:
DMSENUSR

Severity:
ERROR

10211

Explanation:
Error obtaining UID or GID. Insufficient storage

Module:
DMSENUSR

Severity:
ERROR

10212

Explanation:
Error obtaining UID or GID. User not authorized

Module:
DMSENUSR

Severity:
ERROR

10213

Explanation:
Error obtaining UID or GID. CMS internal error

Module:
DMSENUSR

Severity:
ERROR

10214

Explanation:
Error obtaining UID or GID. CP internal error

Module:
DMSENUSR

Severity:
ERROR

10215

Explanation:
Error obtaining UID or GID. Database not available

Module:
DMSENUSR

Severity:
ERROR

10216

Explanation:
Error obtaining UID or GID. User not found

Module:
DMSENUSR

Severity:
ERROR

10217

Explanation:
Error obtaining UID or GID. Group not found

Module:
DMSENUSR

Severity:
ERROR

10220 (1)

Explanation:
File mode other than 1 specified for a BFS file.

Module:
DMSCLBLK

Severity:
WARNING

10220 (2)

Explanation:
File mode value other than 1 specified for a BFS file.

Module:
DMSOPBLK

Severity:
WARNING

10220 (3)

Explanation:
File mode value other than 1 specified for a BFS file.

Module:
DMSOPDBK

Severity:
WARNING

10220 (4)

Explanation:
File mode value other than 1 specified for a BFS file.

Module:
DMSOPEN

Severity:
WARNING

10240

Explanation:
Function not allowed in DOS mode, in SUBSET mode, or on this level of CP.

Module:
DMSENUSR

Severity:
ERROR

20000 (1)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSCLBLK

Severity:
ERROR

20000 (2)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSCLCAT

Severity:
ERROR

20000 (3)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSCLDBK

Severity:
ERROR

20000 (4)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSCLDIR

Severity:
ERROR

20000 (5)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSCLOSE

Severity:
ERROR

20000 (6)

Explanation:
Alias name already exists as a base file or an alias in the same directory.

Module:
DMSCRALI

Severity:
ERROR

20000 (7)

Explanation:
Directory with the same name already exists.

Module:
DMSCRDIR

Severity:
ERROR

20000 (8)

Explanation:
File already exists.

Module:
DMSCRFIL

Severity:
ERROR

20000 (9)

Explanation:
External object or file already exists.

Module:
DMSCROB

Severity:
ERROR

20000 (10)

Explanation:
A specified file space ID is already enrolled.

Module:
DMSENUSR

Severity:
ERROR

20000 (11)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSERASE

Severity:
ERROR

20000 (12)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSEXIDI

Severity:
ERROR

20000 (13)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSEXIFI

Severity:
ERROR

20000 (14)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSEXIST

Severity:
ERROR

20000 (15)

Explanation:
The target file already exists and REPLACE was not specified.

Module:
DMSFILEC

Severity:
ERROR

20000 (16)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSGRANT

Severity:
ERROR

20000 (17)

Explanation:
Intent was NEW, but the file already exists.

Module:
DMSOPBLK

Severity:

ERROR

20000 (18)

Explanation:
Intent was NEW, but the file already exists.

Module:
DMSOPDBK

Severity:
ERROR

20000 (19)

Explanation:
Intent was NEW, but the file already exists.

Module:
DMSOPEN

Severity:
ERROR

20000 (20)

Explanation:
The directory to be relocated already exists in the target directory.

Module:
DMSRELOC

Severity:
ERROR

20000 (21)

Explanation:
The specified 'new' directory name already exists in the specified file pool.

Module:
DMSRENAM

Severity:
ERROR

20000 (22)

Explanation:
Duplicate object found in file pool catalog. This error occurs only when the COMMIT option is specified: you have created a file pool object, and you or another user has concurrently created an object with the same name on another unit of work. The other unit of work was committed first, and your attempt to commit your unit of work has failed.

Module:
DMSREVOK

Severity:
ERROR

20010 (1)

Explanation:
The file to be relocated already exists in the target directory.

Module:
DMSRELOC

Severity:
ERROR

20010 (2)

Explanation:
The file IDs are identical. The source file or directory name is the same as the target.

Module:
DMSRENAM

Severity:
ERROR

30000 (1)

Explanation

You do not have the required authority:

- In an SFS file space, you must have file pool administration authority to perform this work unit *on another user's behalf*. The DMSGETWU routine was used to associate the user ID with the work unit.
- In a BFS file space, you must have file pool administration authority to perform this task.

Module:
Common

Severity:
ERROR

30000 (2)

Explanation:
You do not have the file pool administration authority required to lock a BFS file.

Module:
DMSCRLOC

Severity:
ERROR

30000 (3)

Explanation

You do not have the required authority:

- In an SFS file space, file pool administration authority is required to delete a lock held by another user.
- In a BFS file space, file pool administration authority is required to delete a lock.

Module:
DMSDELOC

Severity:
ERROR

30000 (4)

Explanation:
You do not have the file pool administration authority required to delete a file space.

Module:
DMSDEUSR

Severity:
ERROR

30000 (5)

Explanation:
You do not have the file pool administration authority required to disable a BFS file space.

Module:
DMSDISFS

Severity:
ERROR

30000 (6)

Explanation:
You are not authorized to issue this request or the *owner* specified is not authorized to perform this request.

Module:
DMSDISSG

Severity:
ERROR

30000 (7)

Explanation

You do not have the required authority:

- To enable an SFS file space that was disabled by another user, you must have file pool administration authority.
- To enable a BFS file space, you must have file pool administration authority.

Module:
DMSENAFS

Severity:
ERROR

30000 (8)

Explanation:
You are not a file pool administrator, and the request is to enable a storage group that was disabled by another user.

Module:
DMSENASG

Severity:
ERROR

30000 (9)

Explanation:
You do not have the file pool administration authority required to enroll a file space.

Module:
DMSENUSR

Severity:
ERROR

30000 (10)

Explanation:
You do not have the required authority. File pool administration authority is required to use the CREATEMIG intent or to open a BFS file.

Module:
DMSOPBLK

Severity:
ERROR

30000 (11)

Explanation:
You do not have the required file pool administration authority.

Module:
DMSOPCAT

Severity:
ERROR

30000 (12)

Explanation:
You do not have the file pool administration authority required to issue a request with the GROUP or ALL parameter.

Module:
DMSQFPDS

Severity:
ERROR

30000 (13)

Explanation:
Not authorized: file pool administration authority is required.

Module:
DMSQLIMA

Severity:
ERROR

30000 (14)

Explanation:
Administrator Authority required.

Module:
DMSQUSG

Severity:
ERROR

30000 (15)

Explanation:
You do not have administrator authority.

Module:
DMSQRELBK

Severity:
ERROR

30000 (16)

Explanation:
You do not have file pool administration authority.

Module:
DMSWRACC

Severity:
ERROR

30100

Explanation:
You do not have authority to use the specified file pool. You are not enrolled in the file pool, and PUBLIC is not enrolled.

Module:
Common

Severity:
ERROR

30150

Explanation:
A previous connect occurred for the file pool during this CMS session. User ID is already connected.

Module:
Common

Severity:
WARNING

30180

Explanation:
None of the specified user IDs are connected or no user IDs are connected.

Module:
Common

Severity:
WARNING

30200

Explanation:
NOACCOUNT specified as a file pool server start-up parameter.

Module:
DMSWRACC

Severity:
ERROR

30300

Explanation:
You have attempted a remote connection to a file pool server that was started for local use only.

Module:
Common

Severity:
ERROR

30400

Explanation:
You have attempted a remote connection to a file pool server that was started for SSI use only.

Module:
Common

Severity:
ERROR

30500 (1)

Explanation:
The combination of attributes RECOVER and INPLACE is not supported.

Module:
DMSCATTR

Severity:
ERROR

30500 (2)

Explanation:
RECOVER and INPLACE attributes are mutually exclusive.

Module:
DMSCRFIL

Severity:
ERROR

30500 (3)

Explanation:
The combination of INPLACE and RECOVER is not supported.

Module:
DMSOPBLK

Severity:
ERROR

30500 (4)

Explanation:
System error. The combination of attributes INPLACE and RECOVER is not supported.

Module:
DMSOPDBK

Severity:
ERROR

30500 (5)

Explanation:
The combination of attributes INPLACE and RECOVER is not supported.

Module:
DMSOPEN

Severity:
ERROR

30700 (1)

Explanation:
System error. An attempt was made to update a block in place for a file that is not OPEN for INPLACE processing.

Module:
DMSWRDBK

Severity:
ERROR

30700 (2)

Explanation:
System error. An attempt was made to update a block in place for a file that is not OPEN for INPLACE processing.

Module:
DMSWRITE

Severity:
ERROR

31000 (1)

Explanation:
Mixing recoverable and nonrecoverable updates for the same file is incorrect within a single work unit.

Module:
DMSCRFIL

Severity:
ERROR

31000 (2)

Explanation:
Mixing recoverable and nonrecoverable work for the same file is incorrect within a single work unit.

Module:
DMSFILEC

Severity:
ERROR

31000 (3)

Explanation:
Mixing recoverable and nonrecoverable updates for the same file is incorrect within a single work unit.

Module:
DMSOPBLK

Severity:
ERROR

31000 (4)

Explanation:
Mixing recoverable and nonrecoverable work for the same file is incorrect within a single work unit.

Module:
DMSOPDBK

Severity:
ERROR

31000 (5)

Explanation:
Mixing recoverable and nonrecoverable work for the same file is incorrect within a single work unit.

Module:
DMSOPEN

Severity:
ERROR

31000 (6)

Explanation:
You cannot mix recoverable and nonrecoverable work for the same file within a single work unit. The file is nonrecoverable and has been changed without being committed. To truncate the file with DMSTRUNC, first close and commit it.

Module:
DMSTRUNC

Severity:
ERROR

32000 (1)

Explanation:
Specified file space ID is not enrolled in the file pool.

Module:
DMSOPCAT

Severity:
ERROR

32000 (2)

Explanation:
User ID is not enrolled in the file pool.

Module:
DMSQFPDS

Severity:
ERROR

32010 (1)

Explanation:
The *filespaceid* part of the *filepoolid* **:***filespaceid* parameter is missing or is longer than 8 characters.

Module:
DMSDISFS

Severity:
ERROR

32010 (2)

Explanation:
The *filespaceid* part of the *filepoolid* **:***filespaceid* parameter is missing or is longer than 8 characters, or the *userid* parameter is longer than 8 characters.

Module:
DMSENAFS

Severity:
ERROR

32010 (3)

Explanation:
The user ID parameter is longer than 8 characters.

Module:
DMSENASG

Severity:
ERROR

32010 (4)

Explanation:
The *filespaceid* part of the file space identifier parameter is missing or is longer than 8 characters.

Module:
DMSOPCAT

Severity:
ERROR

32010 (5)

Explanation:
The user ID parameter is missing or is longer than 8 characters.

Module:
DMSQFPDS

Severity:
ERROR

32040

Explanation:
You attempted to delete a file space that is not enrolled.

Module:
DMSDEUSR

Severity:
WARNING

32050 (1)

Explanation:
No file spaces are enrolled in the file pool.

Module:
DMSQLIMA

Severity:
WARNING

32050 (2)

Explanation:
The specified file space ID is not enrolled.

Module:
DMSQLIMU

Severity:
WARNING

32070

Explanation:
Specified DOID is not found or the requestor is not authorized.

Module:
Common

Severity:
WARNING

32080

Explanation:
No lock on the specified type of object exists.

Module:
DMSQFPDS

Severity:
WARNING

32100

Explanation:
A specified file space cannot be deleted because the file space is currently being accessed.

Module:
DMSDEUSR

Severity:
ERROR

32610

Explanation:
One or more specified user IDs have neither READ/WRITE nor NEWREAD/NEWWRITE authority on the object, but a REVOKE AUTHORITY was requested that implies that these authorities existed.

Module:
DMSREVOK

Severity:
WARNING

32620

Explanation:
One or more specified user IDs does not have NEWREAD or NEWWRITE authority on the object.

Module:
DMSREVOK

Severity:
WARNING

32630

Explanation:
One or more specified user IDs does not have DIRREAD or DIRWRITE authority on the object.

Module:
DMSREVOK

Severity:
WARNING

32640

Explanation:
One or more specified user IDs does not have READ or WRITE authority to the specified file pool object.

Module:
DMSREVOK

Severity:
WARNING

32650

Explanation:
READ authority granted to a user ID that already has WRITE authority.

Module:
DMSGRANT

Severity:
WARNING

32660

Explanation:
NEWREAD authority granted to a user ID that already has NEWWRITE authority.

Module:
DMSGRANT

Severity:
WARNING

32670

Explanation:
DIRREAD authority granted to a user ID that already has DIRWRITE authority.

Module:
DMSGRANT

Severity:
WARNING

32680

Explanation:
One or more specified user IDs is your userid. You cannot revoke authority from yourself.

Module:
DMSREVOK

Severity:
WARNING

32690

Explanation:
The default for the DMSGRANT routine (READ or DIRREAD) caused DIRREAD authority to be granted on a directory control directory.

Module:
DMSGRANT

Severity:
WARNING

35000 (1)

Explanation:
The work unit was rolled back. System limits were exceeded when attempting to commit. CMS cannot commit more than approximately 230 resources on a work unit.

Module:
DMSCOMM

Severity:
ERROR

35000 (2)

Explanation:
The work unit was rolled back. System limits were exceeded when attempting to commit. CMS cannot commit more than approximately 230 resources on a work unit.

Module:
DMSRETWU

Severity:
ERROR

40000

Explanation:
An internal SFS limit has been exceeded.

Module:
Common

Severity:
ERROR

44000 (1)

Explanation:
The file does not exist or you do not have the authority to change its attributes.

Module:
DMSCATTR

Severity:
ERROR

44000 (2)

Explanation

Attempt to create an alias failed because:

- The base file, its parent directory, or the target directory does not exist.
- You do not have explicit read or write authority to the base file.
- You do not have write authority to the target directory.
- The owner of the target directory does not have read or write authority to the base file.

Module:
DMSCRALI

Severity:
ERROR

44000 (3)

Explanation:
Parent directory does not exist or you are not the owner of it.

Module:
DMSCRDIR

Severity:
ERROR

44000 (4)

Explanation:
Directory not found, or issuer is not authorized to write to it.

Module:
DMSCRFIL

Severity:
ERROR

44000 (5)

Explanation:
The file pool object to be locked does not exist or you are not authorized to lock it in the requested mode.

Module:
DMSCRLOC

Severity:
ERROR

44000 (6)

Explanation:
Access is not authorized or directory does not exist.

Module:
DMSCROB

Severity:
ERROR

44000 (7)

Explanation:
The specified file pool object does not exist or you are not authorized for it.

Module:
DMSDELOC

Severity:
ERROR

44000 (8)

Explanation:
Directory does not exist or you are not authorized.

Module:
DMSDIRAT

Severity:
ERROR

44000 (9)

Explanation:
The file space to be locked does not exist or you are not authorized to it.

Module:
DMSDISFS

Severity:
ERROR

44000 (10)

Explanation:
The file space does not exist or you are not authorized to enable it.

Module:
DMSENAFS

Severity:
ERROR

44000 (11)

Explanation:
The storage group does not exist or you are not authorized to enable it.

Module:
DMSENASG

Severity:
ERROR

44000 (12)

Explanation:
The specified directory or file does not exist or you are not authorized to erase it.

Module:
DMSERASE

Severity:
ERROR

44000 (13)

Explanation:
The specified directory does not exist or you are not authorized for it.

Module:
DMSEXIDI

Severity:
ERROR

44000 (14)

Explanation:
The specified file, directory, or external object does not exist or you are not authorized for it.

Module:
DMSEXIFI

Severity:
ERROR

44000 (15)

Explanation:
The file, alias, external object, or directory specified by the *uniqueid_fpid* parameter has not been committed to the file pool, or you are not authorized to use it.

Module:
DMSEXIST

Severity:
ERROR

44000 (16)

Explanation:
File does not exist or you are not authorized to grant authority to it.

Module:
DMSGRANT

Severity:
ERROR

44000 (17)

Explanation

- Intent was NEW or CREATEMIG, and the directory does not exist.
- Intent was NEW, and you are not authorized to create a file in the directory.
- Intent was WRITE or REPLACE, and the file does not exist and you are not authorized to create a file in the directory.
- Intent was WRITE or REPLACE, and you are not authorized to write to the file.
- Intent was WRITE or REPLACE, and the directory does not exist.
- Intent was READ, and the file does not exist or you are not authorized to read the file.

Module:
DMSOPBLK

Severity:
ERROR

44000 (18)

Explanation:
Specified file space does not exist, or you are not the owner of the file space and you do not have file pool administration authority.

Module:
DMSOPCAT

Severity:
ERROR

44000 (19)

Explanation

- Intent was NEW, and the directory does not exist or you are not authorized to create a file in the directory, **or**,
- Intent was REPLACE, and the file does not exist and you are not authorized to create a file in the directory, or you are not authorized to write to the file, or the directory does not exist, **or**,
- Intent was READ, and the file does not exist or you are not authorized to read the file.

Module:
DMSOPDBK

Severity:

ERROR

44000 (20)

Explanation:
For Open Directory for AUTH, LOCK, ALIAS, or FILEEXT, the specified directory or specified file does not exist or you are not authorized to read the directory.

Module:
DMSOPDIR

Severity:
ERROR

44000 (21)

Explanation

- Intent was NEW, and the directory does not exist or you are not authorized to create a file in the directory, **or**,
- Intent was WRITE or REPLACE, and the file does not exist and you are not authorized to create a file in the directory, or you are not authorized to write to the file, or the directory does not exist, **or**,
- Intent was READ, and the file does not exist or you are not authorized to read the file.

Module:
DMSOPEN

Severity:
ERROR

44000 (22)

Explanation:
The specified directory does not exist or you are not authorized for it.

Module:
DMSEXIDI

Severity:
ERROR

44000 (23)

Explanation:
Specified storage group does not exist.

Module:
DMSQUSG

Severity:
ERROR

44000 (24)

Explanation:
The directory or file to be relocated does not exist or you are not authorized to relocate it.

Module:
DMSRELOC

Severity:
ERROR

44000 (25)

Explanation:
The directory, alias, external object, or file to be renamed does not exist or you are not authorized to rename it.

Module:
DMSRENAM

Severity:
ERROR

44000 (26)

Explanation:
Specified file or parent directory does not exist or you are not authorized to revoke authorities granted on the file.

Module:
DMSREVOK

Severity:
ERROR

44000 (27)

Explanation:
The file does not exist or you are not authorized to write to the file, or the directory does not exist.

Module:
DMSTRUNC

Severity:
ERROR

44010

Explanation:
Object is protected by external security manager (revoke successful).

Module:
DMSREVOK

Severity:
WARNING

44030 (1)

Explanation:
Intent was WRITE or REPLACE, and the file did not previously exist.

Module:
DMSOPBLK

Severity:
WARNING

44030 (2)

Explanation:
Intent was REPLACE, and the file did not previously exist.

Module:
DMSOPDBK

Severity:
WARNING

44030 (3)

Explanation:
Intent was WRITE or REPLACE, and the file did not previously exist.

Module:
DMSOPEN

Severity:
WARNING

44035 (1)

Explanation:
Intent was REPLACE, and the file did not previously exist. In addition, ALLOWEMPTY was specified but the file pool or minidisk does not support this option.

Module:
DMSOPDBK

Severity:
WARNING

44035 (2)

Explanation:
Intent was WRITE or REPLACE, and the file did not previously exist. In addition, ALLOWEMPTY was specified but the file pool or minidisk does not support this option.

Module:
DMSOPEN

Severity:
WARNING

44040 (1)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDA

Severity:
WARNING

44040 (2)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDD

Severity:
WARNING

44040 (3)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDF

Severity:
WARNING

44040 (4)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDI

Severity:
WARNING

44040 (5)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDK

Severity:
WARNING

44040 (6)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDL

Severity:
WARNING

44040 (7)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDS

Severity:
WARNING

44040 (8)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDT

Severity:
WARNING

44040 (9)

Explanation:
All of the selected data has been returned. Subsequent Get Directory requests will result in reason code 90275.

Module:
DMSGETDX

Severity:
WARNING

44040 (10)

Explanation:
All data has previously been returned. The error data buffer is unchanged.

Module:
DMSGETER

Severity:
WARNING

44040 (11)

Explanation:
All data has previously been returned. The *error_data_buffer* buffer is unchanged.

Module:
DMSGETSP

Severity:
WARNING

44040 (12)

Explanation:
No more entries to follow.

Module:
DMSRDCAT

Severity:
WARNING

44060

Explanation:
The source file does not exist or you are not authorized to read it.

Module:
DMSFILEC

Severity:
ERROR

44100 (1)

Explanation:
You are not authorized to create the directory named on this request. Note, this error only occurs when SFS is running with an external security manager.

Module:
DMSCRDIR

Severity:
ERROR

44100 (2)

Explanation:
You are not authorized to erase one or more objects in the directory.

Module:
DMSERASE

Severity:
ERROR

44100 (3)

Explanation

One of the following is true:

- Target directory does not exist
- Target file does not exist, and you are not authorized to create a file in the target directory
- Target file exists, but you are not authorized to write to it

Module:
DMSFILEC

Severity:
ERROR

44100 (4)

Explanation:
The target directory does not exist or you are not authorized to write to it.

Module:
DMSRELOC

Severity:
ERROR

44100 (5)

Explanation:
You are not authorized to erase one or more objects in the directory.

Module:
DMSERASE

Severity:
ERROR

44100 (6)

Explanation:
You are not authorized to create the directory named on this request. Note, this error only occurs when SFS is running with an external security manager.

Module:
DMSCRDIR

Severity:
ERROR

44150

Explanation:
REPLACE was specified, but the target file did not previously exist. The target file was created.

Module:
DMSFILEC

Severity:
WARNING

44200 (1)

Explanation:
The file is already open for WRITE, NEW, or REPLACE using DMSOPEN or DMSOPDBK.

Module:
DMSFILEC

Severity:
ERROR

44200 (2)

Explanation:
Your attempt to open an SFS file with an intent of NEW, WRITE, REPLACE, or CREATEMIG failed because you already have the file open with one of these intents on a DMSOPEN, DMSOPBLK, or DMSOPDBK request. This reason code is returned only if both attempts to open the file occurred on the same work unit.

Module:
DMSOPBLK

Severity:
ERROR

44200 (3)

Explanation
Your attempt to open a file failed because you already have it open with DMSOPEN, DMSOPBLK, or DMSOPDBK:

- You tried to open an SFS or minidisk file for NEW or REPLACE and you already have it opened for NEW, WRITE, or REPLACE.
- Or you tried to open a minidisk file for REPLACE and you already have it opened for READ.
- Or you tried to open a minidisk file for READ and you already have it opened for NEW, WRITE, or REPLACE.

This reason code is returned for SFS files only if both attempts to open the file occurred on the same work unit.

Module:
DMSOPDBK

Severity:
ERROR

44200 (4)

Explanation
Your attempt to open a file failed because you already have it open with DMSOPEN, DMSOPBLK, or DMSOPDBK:

- You tried to open an SFS or minidisk file for NEW, WRITE, or REPLACE and you had already opened it for NEW, WRITE, or REPLACE.
- Or you tried to open a minidisk file for WRITE or REPLACE and you had already opened it for READ.
- Or you tried to open a minidisk file for READ and you had already opened it for NEW, WRITE, or REPLACE.

This reason code is returned for SFS files only if both attempts to open the file occurred on the same work unit.

Module:
DMSOPEN

Severity:
ERROR

44200 (5)

Explanation:
The file is already open for WRITE, NEW, or REPLACE using DMSOPEN or DMSOPDBK.

Module:
DMSTRUNC

Severity:
ERROR

44300

Explanation:
Grant failed because object is protected by external security manager.

Module:
DMSGRANT

Severity:
ERROR

44400

Explanation:
The external security manager (ESM) returned an error to the server machine for this routine. Refer to your ESM documentation to determine why the ESM returned an error to the server (perhaps the *userdata* was not correct). This reason code is also returned if the external security manager does not support ESM user data.

Module:
DMSUDATA

Severity:
ERROR

44450

Explanation:
External Security Manager (ESM) is active for the file pool server.

Module:
Common

Severity:
WARNING

44700 (1)

Explanation:
Input file has been move and there is no active DFSMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)

Module:
DMSOPBLK

Severity:
ERROR

44700 (2)

Explanation:
There is no active external security manager. (The file pool server is running with the NOESECURITY start-up parameter in effect.)

Module:
DMSUDATA

Severity:
ERROR

44900

Explanation:
The catalog storage group contains no data.

Module:
DMSRDCAT

Severity:
ERROR

50100 (1)

Explanation:
Specified storage group number is invalid (less than 2 or greater than MAXDISKS server parameter).

Module:
DMSDISSG

Severity:
ERROR

50100 (2)

Explanation:
Incorrect storage group specified.

Module:
DMSENASG

Severity:
ERROR

50100 (3)

Explanation:
Specified storage group number is invalid (less than 1 or greater than MAXDISKS server parameter).

Module:
DMSENUSR

Severity:
ERROR

50100 (4)

Explanation:
Specified storage group number is not valid (less than 2 or greater than MAXDISKS server parameter).

Module:
DMSOPCAT

Severity:
ERROR

50100 (5)

Explanation:
Incorrect storage group number specified. It cannot be less than 2 or greater than the MAXDISKS server parameter.

Module:
DMSQFPDS

Severity:
ERROR

50100 (6)

Explanation:
Specified storage group number is not valid (less than 2 or greater than MAXDISKS server parameter).

Module:
DMSRELBK

Severity:
ERROR

50103 (1)

Explanation:
Storage group number is not valid. Number cannot be 1 or less than zero.

Module:
DMSOPCAT

Severity:
ERROR

50103 (2)

Explanation:
Incorrect storage group number specified. Storage group cannot be 1 or less than 0.

Module:
DMSQUSG

Severity:
ERROR

50105 (1)

Explanation:
Storage group number is not valid. When opening a directory or a file space, the specified storage group number must be zero.

Module:
DMSOPCAT

Severity:
ERROR

50105 (2)

Explanation:
Incorrect storage group number specified. When the FILESPACE or ALL parameter is specified, the storage group number must be 0.

Module:
DMSQFPDS

Severity:
ERROR

50200 (1)

Explanation:
Specified storage group does not exist.

Module:
DMSDISSG

Severity:
ERROR

50200 (2)

Explanation:
Specified storage group does not exist.

Module:
DMSENUSR

Severity:
ERROR

50200 (3)

Explanation:
Specified storage group does not exist.

Module:
DMSOPCAT

Severity:
ERROR

50200 (4)

Explanation:
Specified storage group does not exist.

Module:
DMSQFPDS

Severity:
ERROR

50200 (5)

Explanation:
Specified storage group does not exist.

Module:
DMSRELBK

Severity:
ERROR

50300

Explanation:
Specified storage group does not exist and MAXDISKS limit reached by file pool server.

Module:
DMSENUSR

Severity:
ERROR

50400

Explanation:
You cannot assign a file space to storage group 1.

Module:
DMSENUSR

Severity:
ERROR

50500 (1)

Explanation:
The operation was successful but the work could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in

one of the FPERROR entries. See "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCLBLK

Severity:
ERROR

50500 (2)

Explanation:
Attempt to exceed the number of 4KB file blocks allowed for this user. The processing requested by this routine was performed, but has not been committed. Your application must either take remedial action (such as erase some other files in the affected file space) and then call DMSCOMM for this work unit, or call DMSROLLB to roll back this work unit.

Module:
DMSCLDBK

Severity:
ERROR

50500 (3)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCLDIR

Severity:
ERROR

50500 (4)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCLOSE

Severity:
ERROR

50500 (5)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the commitment failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCRALI

Severity:
ERROR

50500 (6)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the commitment failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCRDIR

Severity:
ERROR

50500 (7)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCRFIL

Severity:
ERROR

50500 (8)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSCROB

Severity:
ERROR

50500 (9)

Explanation:

The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSERASE

Severity:
ERROR

50500 (10)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSEXIDI

Severity:
ERROR

50500 (11)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSEXIFI

Severity:
ERROR

50500 (12)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSEXIST

Severity:
ERROR

50500 (13)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSFILEC

Severity:
ERROR

50500 (14)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSGRANT

Severity:
ERROR

50500 (15)

Explanation:
Attempt to exceed the maximum number of 4KB file space blocks allowed for this user. Applicable only when NEW, WRITE, REPLACE, or CREATEMIG parameter is specified.

Module:
DMSOPBLK

Severity:
ERROR

50500 (16)

Explanation:
Attempt to exceed the maximum number of 4KB file space blocks allowed for this user. Applicable only when NEW or REPLACE option is specified.

Module:
DMSOPDBK

Severity:
ERROR

50500 (17)

Explanation:
Attempt to exceed the maximum number of 4KB file space blocks allowed for this user. Applicable

only when NEW, WRITE, or REPLACE parameter is specified.

Module:
DMSOPEN

Severity:
ERROR

50500 (18)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSQOBJ

Severity:
ERROR

50500 (19)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSRENAM

Severity:
ERROR

50500 (20)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSREVOK

Severity:
ERROR

50500 (21)

Explanation:
The operation was successful but the work unit could not be committed. If the *wuerror* parameter was specified, a code for the specific reason that the attempt to commit failed is put in the *error_reascode_info* field in one of the FPERROR entries. See the "Return Codes and Reason Codes" on page 73 for descriptions of these codes.

Module:
DMSTRUNC

Severity:
ERROR

50600 (1)

Explanation:
The user has no storage blocks allocated to his file space.

Module:
DMSCRFIL

Severity:
ERROR

50600 (2)

Explanation:
File cannot be copied. The user has no storage blocks allocated to his file space.

Module:
DMSFILEC

Severity:
ERROR

50600 (3)

Explanation:
File cannot be opened. The user has no storage blocks allocated to his file space.

Module:
DMSOPBLK

Severity:
ERROR

50600 (4)

Explanation:
File cannot be opened. The user has no storage blocks allocated to his file space.

Module:
DMSOPEN

Severity:
ERROR

50700 (1)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCLBLK

Severity:

ERROR

50700 (2)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCLDBK

Severity:
ERROR

50700 (3)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCLDIR

Severity:
ERROR

50700 (4)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCLOSE

Severity:
ERROR

50700 (5)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCRALI

Severity:
ERROR

50700 (6)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCRDIR

Severity:
ERROR

50700 (7)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCRFIL

Severity:
ERROR

50700 (8)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSCROB

Severity:
ERROR

50700 (9)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSERASE

Severity:
ERROR

50700 (10)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSEXIDI

Severity:
ERROR

50700 (11)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSEXIFI

Severity:
ERROR

50700 (12)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSEXIST

Severity:
ERROR

50700 (13)

Explanation:

There is no room in the file space to complete this request.

Module:
DMSFILEC

Severity:
ERROR

50700 (14)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSGRANT

Severity:
ERROR

50700 (15)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSQOBJ

Severity:
ERROR

50700 (16)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSRENAM

Severity:
ERROR

50700 (17)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSREVOK

Severity:
ERROR

50700 (18)

Explanation:
There is no room in the file space to complete this request.

Module:
DMSTRUNC

Severity:
ERROR

51000 (1)

Explanation:
Storage group space limit exceeded. Applicable only when COMMIT parameter is specified.

Module:
DMSCLBLK

Severity:
ERROR

51000 (2)

Explanation:
Storage group space limit exceeded.

Module:
DMSCLDBK

Severity:
ERROR

51000 (3)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSCLDIR

Severity:
ERROR

51000 (4)

Explanation:
Storage group space limit exceeded.

Module:
DMSCLOSE

Severity:
ERROR

51000 (5)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSCRALI

Severity:
ERROR

51000 (6)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSCRDIR

Severity:
ERROR

51000 (7)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSERASE

Severity:
ERROR

51000 (8)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSEXIDI

Severity:
ERROR

51000 (9)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSEXIFI

Severity:
ERROR

51000 (10)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSEXIST

Severity:
ERROR

51000 (11)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSFILEC

Severity:
ERROR

51000 (12)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSGRANT

Severity:
ERROR

51000 (13)

Explanation:
Storage group space limit exceeded.

Module:
DMSRDDBK

Severity:
ERROR

51000 (14)

Explanation:
Storage group space limit exceeded.

Module:
DMSREAD

Severity:
ERROR

51000 (15)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSRENAM

Severity:
ERROR

51000 (16)

Explanation:
COMMIT was specified. Storage group space limit exceeded.

Module:
DMSREVOK

Severity:
ERROR

51000 (17)

Explanation:
Storage group space limit exceeded.

Module:
DMSWRBLK

Severity:
ERROR

51000 (18)

Explanation:
Storage group space limit exceeded.

Module:
DMSWRDBK

Severity:
ERROR

51000 (19)

Explanation:
Storage group space limit exceeded.

Module:
DMSWRITE

Severity:
ERROR

51010

Explanation:
No space for data left in catalog space.

Module:
Common

Severity:
ERROR

51020

Explanation:
No index space left in catalog space.

Module:
Common

Severity:
ERROR

51050 (1)

Explanation:
File space warning threshold reached or exceeded.

Module:
DMSCLBLK

Severity:
WARNING

51050 (2)

Explanation:
File space warning threshold reached or exceeded.

Module:
DMSCLDBK

Severity:
WARNING

51050 (3)

Explanation:
File space warning threshold reached or exceeded.

Module:
DMSCLOSE

Severity:
WARNING

51050 (4)

Explanation:
Still exceeding file space warning threshold after erasing the file or directory.

Module:
DMSERASE

Severity:
WARNING

51050 (5)

Explanation:
File space warning threshold reached or exceeded.

Module:
DMSFILEC

Severity:
WARNING

51050 (6)

Explanation:
Call to DMSRDDBK was successful, but your file space warning threshold was reached or exceeded. This can occur on a read request because of CMS buffering.

Module:
DMSRDDBK

Severity:
WARNING

51050 (7)

Explanation:
Call to DMSREAD was successful, but your file space warning threshold was reached or exceeded. This can occur on a read request because of CMS buffering.

Module:
DMSREAD

Severity:
WARNING

51050 (8)

Explanation:
The file-space-warning threshold was reached or exceeded.

Module:
DMSWRBLK

Severity:
WARNING

51050 (9)

Explanation:
File space warning threshold reached or exceeded.

Module:
DMSWRDBK

Severity:
WARNING

51050 (10)

Explanation:
The file-space-warning threshold was reached or exceeded.

Module:
DMSWRITE

Severity:
WARNING

51051 (1)

Explanation:
You specified a nonblank value for *create_date* or *create_time* and opened an existing file with an intent of REPLACE or WRITE. You cannot change the creation date or time of an existing file.

Module:
DMSOPBLK

Severity:
WARNING

51051 (2)

Explanation:
You specified nonblank values for *create_date* or *create_time* and opened an existing file with an intent of REPLACE. You cannot change the creation date or creation time of an existing file.

Module:
DMSOPDBK

Severity:
WARNING

51051 (3)

Explanation:
You specified nonblank values for *create_date* or *create_time* and opened an existing file with an intent of WRITE or REPLACE. You cannot change the creation date or creation time of an existing file.

Module:
DMSOPEN

Severity:
WARNING

51055

Explanation:
Specified warning threshold percentage is invalid.

Module:
DMSENUSR

Severity:
ERROR

51060 (1)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSCLBLK

Severity:
WARNING

51060 (2)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit processing.

Module:
DMSCLDBK

Severity:
WARNING

51060 (3)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSCLDIR

Severity:
WARNING

51060 (4)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit processing.

Module:
DMSCLOSE

Severity:
WARNING

51060 (5)

Explanation:
The user reached or exceeded the SFS file space threshold for one or more file spaces.

Module:
DMSCOMM

Severity:
WARNING

51060 (6)

Explanation:
The file space warning threshold was reached or exceeded for one or more file spaces while committing or rolling back a work unit.

Module:
DMSCRALI

Severity:
WARNING

51060 (7)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSCRDIR

Severity:
WARNING

51060 (8)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit processing.

Module:
DMSERASE

Severity:
WARNING

51060 (9)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces while committing or rolling back a work unit.

Module:
DMSEXIDI

Severity:
WARNING

51060 (10)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSEXIFI

Severity:
WARNING

51060 (11)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSEXIST

Severity:
WARNING

51060 (12)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSFILEC

Severity:
WARNING

51060 (13)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSGRANT

Severity:
WARNING

51060 (14)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSRENAM

Severity:
WARNING

51060 (15)

Explanation:
The user exceeded the SFS file space threshold for one or more file spaces.

Module:
DMSRETWU

Severity:
WARNING

51060 (16)

Explanation:
File space warning threshold reached or exceeded for one or more file spaces during commit or rollback processing.

Module:
DMSREVOK

Severity:
WARNING

51060 (17)

Explanation:
The user exceeded the SFS file space threshold for one or more file spaces. (This occurred when closing nonrecoverable files.)

Module:
DMSROLLB

Severity:
WARNING

51100 (1)

Explanation:
System error. No minidisks assigned to the storage group. Applicable only when COMMIT parameter is specified.

Module:
DMSCLBLK

Severity:
ERROR

51100 (2)

Explanation:
System error. No minidisks assigned to the storage group.

Module:
DMSCLDBK

Severity:
ERROR

51100 (3)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSCLDIR

Severity:
ERROR

51100 (4)

Explanation:
System error. No minidisks assigned to the storage group.

Module:
DMSCLOSE

Severity:
ERROR

51100 (5)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSCRALI

Severity:
ERROR

51100 (6)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSCRDIR

Severity:
ERROR

51100 (7)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSERASE

Severity:
ERROR

51100 (8)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSEXIDI

Severity:
ERROR

51100 (9)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSEXIFI

Severity:
ERROR

51100 (10)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSEXIST

Severity:
ERROR

51100 (11)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSFILEC

Severity:
ERROR

51100 (12)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSGRANT

Severity:
ERROR

51100 (13)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSRENAM

Severity:
ERROR

51100 (14)

Explanation:
System error. COMMIT was specified. No minidisks assigned to the storage group.

Module:
DMSREVOK

Severity:
ERROR

54000 (1)

Explanation:
Attempt to read logical block number not associated with the file.

Module:
DMSRDBLK

Severity:
ERROR

54000 (2)

Explanation:
System error. Attempt to read logical block number not associated with the file.

Module:
DMSRDDBK

Severity:
ERROR

54000 (3)

Explanation:
System error. Attempt to read logical block number not associated with the file.

Module:
DMSREAD

Severity:
ERROR

54000 (4)

Explanation:
System error. Attempt to read logical block number that is not associated with the file. This can occur on a write request due to CMS buffering.

Module:
DMSWRDBK

Severity:
ERROR

54000 (5)

Explanation:
System error. Attempt to read logical block number that is not associated with the file. This can occur on a write request due to CMS buffering.

Module:
DMSWRITE

Severity:
ERROR

54500 (1)

Explanation:
The CRR recovery server log limits were exceeded. The work unit was rolled back.

Module:
DMSCOMM

Severity:
ERROR

54500 (2)

Explanation:
The CRR recovery server log limits were exceeded.

Module:
DMSRETWU

Severity:
ERROR

55000 (1)

Explanation:
Insufficient virtual storage in the file pool server machine to process your request.

Module:
Common

Severity:
ERROR

55000 (2)

Explanation:
Insufficient virtual storage in the CRR recovery server.

Module:
DMSCHREG

Severity:
ERROR

55000 (3)

Explanation:
Insufficient virtual storage in the CRR recovery server.

Module:
DMSGETRS

Severity:
ERROR

55000 (4)

Explanation:
Insufficient virtual storage in the CRR recovery server.

Module:
DMSREG

Severity:
ERROR

55777

Explanation:
The ERASE function issued by DMSCLOSE for a file mode number 3 file failed.

Module:
DMSCLOSE

Severity:
WARNING

56000

Explanation:
Request denied because you have a file pool catalog open for WRITE for the specified work unit ID. The only requests that will be accepted for that work unit ID are Write Catalog, Close Catalog, and Rollback.

Module:
Common

Severity:
ERROR

56100

Explanation:
Required explicit lock not in effect. Prior to the Open Catalog request, the object being opened must have been locked SHARE or higher for READ, or EXCLUSIVE for WRITE.

Module:
DMSOPCAT

Severity:
ERROR

56300

Explanation:
Invalid record type.

Module:
DMSWRCAT

Severity:
ERROR

56400 (1)

Explanation:
System error. The catalog is not open.

Module:
DMSCLCAT

Severity:
ERROR

56400 (2)

Explanation:
Catalog is not open, or is not open with intent READ, READEXT or FILEATTR.

Module:
DMSRDCAT

Severity:
ERROR

56400 (3)

Explanation:
Catalog is not open, or the requested operation conflicts with the open intent.

Module:
DMSWRCAT

Severity:
ERROR

57000

Explanation:
The storage group to which this user belongs is being restored.

Module:
DMSQSFSL

Severity:
ERROR

57050

Explanation:
You attempted to restore a user ID that is currently enrolled in other storage groups. Such user IDs are returned in a list as described in Close Catalog output. The file spaces of the listed user IDs were not restored into the storage group.

Module:
DMSCLCAT

Severity:

WARNING

57080

Explanation:
User IDs were deleted from the storage group. If a user ID was found in the storage group but not in the restore file, the file space of the user ID was deleted from the storage group. These user IDs are returned in a list as described by Close Catalog output.

Module:
DMSCLCAT

Severity:
WARNING

Reason Codes 60000-86400 generated by the VMLIB CSL routines

Explanation, originating module, and severity is documented for reason codes 60000 - 86400, which are generated by VMLIB CSL routines.

60000

Explanation:
The specified directory contains files, aliases, or external objects, and the FILES option was not specified.

Module:
DMSERASE

Severity:
ERROR

60100 (1)

Explanation:
One or more files are open in the directory. Applicable only when the directory has the DIRCONTROL attribute and is not in a data space.

Module:
DMSCLDIR

Severity:
ERROR

60100 (2)

Explanation:
You have the specified file or one or more files in the specified directory open.

Module:
DMSERASE

Severity:
ERROR

60100 (3)

Explanation:
You have the directory or file open.

Module:
DMSRENAM

Severity:
ERROR

60200 (1)

Explanation:
Attempt to create top-level directory.

Module:
DMSCRDIR

Severity:
ERROR

60200 (2)

Explanation:
You cannot erase a top-level directory.

Module:
DMSERASE

Severity:
ERROR

60200 (3)

Explanation:
You have attempted to relocate a top-level directory.

Module:
DMSRELOC

Severity:
ERROR

60200 (4)

Explanation:
You have attempted to rename a top-level directory.

Module:
DMSRENAM

Severity:
ERROR

60300

Explanation:
Specified directory contains one or more subdirectories.

Module:
DMSERASE

Severity:
ERROR

60400

Explanation:
You have the directory open.

Module:

DMSERASE

Severity:
ERROR

61000

Explanation:
Your Relocate request would have resulted in a directory name having more than eight levels.

Module:
DMSRELOC

Severity:
ERROR

61200

Explanation:
The parent of the directory to be relocated has the same name as the specified target directory.

Module:
DMSRELOC

Severity:
ERROR

61300

Explanation:
The directory to be relocated has the same name as the specified target directory.

Module:
DMSRELOC

Severity:
ERROR

61400

Explanation:
You have attempted to relocate a directory such that the directory would be a subdirectory of itself.

Module:
DMSRELOC

Severity:
ERROR

61500

Explanation:
The directory to be relocated and the target directory are not in the same directory hierarchy.

Module:
DMSRELOC

Severity:
ERROR

61600

Explanation:
Attempt to rename other than the lowest level of a directory name.

Module:
DMSRENAM

Severity:
ERROR

61610

Explanation:
Invalid file mode change.

Module:
DMSRENAM

Severity:
ERROR

61620

Explanation:
An unresolved alias was created.

Module:
DMSCRALI

Severity:
WARNING

61621

Explanation:
An unresolved alias cannot be resolved.

Module:
Common

Severity:
WARNING

61700

Explanation:
Attempt to rename a file to a directory or a directory to a file.

Module:
DMSRENAM

Severity:
ERROR

61800

Explanation:
Attempt was made to change file mode number of an alias. The file mode number of the alias remains the same as the file mode number of the base file.

Module:
DMSRENAM

Severity:
ERROR

61801

Explanation:

The directory name was not specified on the command, and no DIRCONTL directories are accessed.

Module:
Common

Severity:
WARNING

61803 (1)

Explanation:
Attempt to set the DIRCONTL attribute to its current setting.

Module:
Common

Severity:
WARNING

61803 (2)

Explanation:
The directory attribute was not changed. For example, you specified FILECONTROL for a file control directory.

Module:
DMSDIRAT

Severity:
WARNING

61804

Explanation:
There are no directories assigned data space eligibility in this file pool.

Module:
Common

Severity:
WARNING

61900

Explanation:
You cannot change file mode numbers when renaming aliases

Module:
DMSRENAM

Severity:
ERROR

62600

Explanation:
DIRCONTROL parameter specified without also specifying FORCE, and individual authorities exist on the directory or on files within it.

Module:
DMSDIRAT

Severity:
ERROR

62800

Explanation:
Explicit locks are held on the directory or files in the directory.

Module:
DMSDIRAT

Severity:
ERROR

62900

Explanation:
The directory was accessed or in use by an administrator or the owner, when you tried to change the directory attribute from FILECONTROL to DIRCONTROL without specifying FORCE.

Module:
DMSDIRAT

Severity:
ERROR

63000

Explanation:
You could not change the directory attribute from DIRCONTROL to FILECONTROL because either you had the directory accessed R/O or someone else had it accessed R/W.

Module:
DMSDIRAT

Severity:
ERROR

63100

Explanation:
DIRREAD or DIRWRITE authority cannot be specified on a file control directory.

Module:
DMSGRANT

Severity:
ERROR

63200

Explanation:
DIRREAD or DIRWRITE or NEWREAD or NEWWRITE or READDIRREAD authority parameter cannot be specified for a file.

Module:
DMSGRANT

Severity:
ERROR

63300

Explanation:

READ, WRITE, NEWREAD, or NEWWRITE was specified for a directory control directory or a file in a directory control directory.

Module:
DMSGRANT

Severity:
ERROR

63400

Explanation:
KEEPREAD, KEEPNEWREAD, NEWAUTH, RWAUTH, or *fn_ft* specified for a directory control directory.

Module:
DMSREVOK

Severity:
ERROR

63500

Explanation:
KEEPDIRREAD specified on a file control directory.

Module:
DMSREVOK

Severity:
ERROR

63600

Explanation:
KEEPDIRREAD, KEEPNEWREAD, or NEWAUTH specified on a base file or alias.

Module:
DMSREVOK

Severity:
ERROR

63700 (1)

Explanation:
The specified file resides in a directory control directory that is accessed read-only.

Module:
DMSCATTR

Severity:
ERROR

63700 (2)

Explanation:
Specified parent directory control directory is accessed read-only.

Module:
DMSCRDIR

Severity:
ERROR

63700 (3)

Explanation:
The directory is a directory control directory, accessed R/O by the issuer.

Module:
DMSCRFIL

Severity:
ERROR

63700 (4)

Explanation:
You cannot create an external object in a directory control directory that is accessed as read-only.

Module:
DMSCROB

Severity:
ERROR

63700 (5)

Explanation:
Specified directory control directory is accessed read-only.

Module:
DMSERASE

Severity:
ERROR

63700 (6)

Explanation:
The specified DIRCONTROL directory is accessed read-only.

Module:
DMSFILEC

Severity:
ERROR

63700 (7)

Explanation:
Directory control directory is accessed read-only.

Module:
DMSOPBLK

Severity:
ERROR

63700 (8)

Explanation:
Directory control directory specified and another user has it locked or accessed R/W, or you have it accessed R/O.

Module:
DMSOPDBK

Severity:
ERROR

63700 (9)

Explanation:
Directory control directory is accessed read-only.

Module:
DMSOPEN

Severity:
ERROR

63700 (10)

Explanation:
Specified directory control directory or its parent is accessed read-only.

Module:
DMSRELOC

Severity:
ERROR

63700 (11)

Explanation:
Specified directory control directory is accessed read-only.

Module:
DMSRENAM

Severity:
ERROR

63700 (12)

Explanation:
Specified directory control directory is accessed read-only.

Module:
DMSREVOK

Severity:
ERROR

63700 (13)

Explanation:
Directory control directory is accessed read-only.

Module:
DMSTRUNC

Severity:
ERROR

63800 (1)

Explanation:
Specified directory is a directory control. Aliases cannot be created in directory control directories.

Module:
DMSCRALI

Severity:
ERROR

63800 (2)

Explanation:
The specified directory is a directory control directory. The CREATEMIG keyword is not supported for directory control directories.

Module:
DMSOPBLK

Severity:
ERROR

63800 (3)

Explanation:
Specified directory is a directory control directory.

Module:
DMSRELOC

Severity:
ERROR

63900

Explanation:
You cannot lock a file in a directory control directory SHARE or EXCLUSIVE.

Module:
DMSCRLOC

Severity:
ERROR

64000

Explanation:
You cannot lock a directory control directory SHARE or EXCLUSIVE.

Module:
DMSCRLOC

Severity:
ERROR

64200

Explanation:
The directory control directory you specified is empty. It has no files or subdirectories.

Module:
DMSOPDIR

Severity:
ERROR

64300 (1)

Explanation:
The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.

Module:
DMSCATTR

Severity:
ERROR

64300 (2)

Explanation:
The specified target file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.

Module:
DMSFILEC

Severity:
ERROR

64300 (3)

Explanation:
The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.

Module:
DMSOPDBK

Severity:
ERROR

64300 (4)

Explanation:
The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.

Module:
DMSOPEN

Severity:
ERROR

64300 (5)

Explanation:
The specified file is an alias whose base file is in a directory control directory that is accessed read-only by the issuer.

Module:
DMSTRUNC

Severity:
ERROR

64400

Explanation:
Aliases exist in the directory and you tried to change the directory attribute to DIRCONTROL without specifying FORCE.

Module:
DMSDIRAT

Severity:
ERROR

64500

Explanation:
Operation failed due to a mismatch between the level of code of your virtual machine and that of the file pool server machine.

Module:
Common

Severity:
ERROR

64600

Explanation:
Internal DFSMS/VM processing error. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the DFSMS/VM error code.

Module:
Common

Severity:
ERROR

64700

Explanation:
File or directory creation or file recall was rejected by a DFSMS/VM ACS routine. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the ACS routine return code.

Module:
Common

Severity:
ERROR

64800

Explanation:
Unexpected error in a DFSMS/VM ACS routine. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the ACS routine return code.

Module:
Common

Severity:
ERROR

64900

Explanation:
Communication error in DFSMS/VM. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the APPC/VM return code.

Module:
Common

Severity:
ERROR

65200 (1)

Explanation:
File is in DFSMS/VM migrated status and implicit RECALL is set to OFF. (Set RECALL ON or issue a DFSMS RECALL command.)

Module:
DMSFILEC

Severity:
ERROR

65200 (2)

Explanation:
File is in DFSMS/VM migrated status and implicit RECALL is set to OFF. (Set RECALL ON or issue a DFSMS RECALL command).

Module:
DMSOPBLK

Severity:
ERROR

65200 (3)

Explanation:
File is migrated and implicit RECALL is set to OFF. Set RECALL ON or issue a DFSMS RECALL command.

Module:
DMSOPDBK

Severity:
ERROR

65200 (4)

Explanation:
File is in DFSMS/VM migrated status and implicit RECALL is set to OFF. (Set RECALL ON or issue a DFSMS RECALL command.)

Module:
DMSOPEN

Severity:
ERROR

65200 (5)

Explanation:
File is migrated and implicit RECALL is set to OFF. Set RECALL ON or issue a DFSMS RECALL command.

Module:
DMSTRUNC

Severity:
ERROR

65300

Explanation:
DFSMS/VM encountered an SFS error while recalling a file. If you specified the *wuerror* parameter, the *error_reascode_info* parameter of DMSWUERR (FPEAUGMT field in the FPERROR macro) returns the SFS reason code.

Module:
Common

Severity:
ERROR

65400 (1)

Explanation:
This operation cannot be performed on the specified type of object (external object, unresolved alias, or BFS object). Or the operation is allowed for BFS objects, but the file ID passed to the routine does not identify a BFS regular file.

Module:
Common

Severity:
ERROR

65400 (2)

Explanation

Request cannot be performed on a BFS object due to one or more of the following:

- *lrecl* value other than 1 specified
- *recform* value other than F specified

Module:
DMSCLBLK

Severity:
ERROR

65400 (3)

Explanation:
Request cannot be performed on a BFS object because a *lrecl* value other than 1 specified.

Module:
DMSCLDBK

Severity:
ERROR

65400 (4)

Explanation:
Object is not a BFS regular file. Only regular BFS files can be locked.

Module:
DMSCRLOC

Severity:
ERROR

65400 (5)

Explanation:
DATAONLY is invalid for an external object, or you tried to erase a BFS object that is not a regular file.

Module:
DMSERASE

Severity:
ERROR

65400 (6)

Explanation:
Specified operation cannot be performed on an external object or on a BFS file.

Module:
DMSFILEC

Severity:
ERROR

65400 (7)

Explanation

Request cannot be performed on a BFS object because of one or more of the following:

- Intent was not READ or REPLACE for a BFS file.
- BFS file has more than 2^{31}-1 records (bytes).
- Intent was REPLACE for a BFS file and the file does not exist.

Module:
DMSOPBLK

Severity:
ERROR

65400 (8)

Explanation

Specified operation cannot be performed on an external object. Request cannot be performed on a BFS object due to one or more of the following:

- Record format other than F specified for a BFS file.
- Intent was not READ or REPLACE for a BFS file.
- BFS file contains more than 2^{31}-1 records (bytes).
- Intent was REPLACE for a BFS file and the file does not exist.

Module:
DMSOPDBK

Severity:
ERROR

65400 (9)

Explanation:
The operation cannot be performed against a BFS object.

Module:
DMSOPDIR

Severity:
ERROR

65400 (10)

Explanation

Specified operation cannot be performed on an external object. The operation cannot be performed against a BFS object due to one or more of the following:

- Record format other than F specified for a BFS file.
- Intent was not READ or REPLACE for a BFS file.
- BFS file has more than 2^{31}-1 records (bytes).
- Intent was REPLACE for a BFS file and the file does not exist.

Module:
DMSOPEN

Severity:
ERROR

65400 (11)

Explanation:
Specified operation cannot be performed on an external object or on a BFS file.

Module:
DMSTRUNC

Severity:
ERROR

65500 (1)

Explanation:
File is in DFSMS/VM migrated and there is no active DFSMS. (The file pool server is running with the NODFSMS startup parameter in effect.)

Module:
DMSFILEC

Severity:
ERROR

65500 (2)

Explanation:
File is in DFSMS/VM migrated status and there is no active DFSMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)

Module:

DMSOPBLK

Severity:
ERROR

65500 (3)

Explanation:
Input file is migrated and there is no active SMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)

Module:
DMSOPDBK

Severity:
ERROR

65500 (4)

Explanation:
File is in DFSMS/VM migrated status and there is no active DFSMS. (The file pool server is running with the NODFSMS startup parameter in effect.)

Module:
DMSOPEN

Severity:
ERROR

65500 (5)

Explanation:
Input file is migrated and there is no active SMS. (The file pool server is running with the NODFSMS start-up parameter in effect.)

Module:
DMSTRUNC

Severity:
ERROR

65700

Explanation:
The directory contains one or more DFSMS/VM migrated files when you attempted to change the directory attribute to DIRCONTROL.

Module:
DMSDIRAT

Severity:
ERROR

65900 (1)

Explanation:
A system error occurred in the file pool server access routine.

Module:
DMSOPDBK

Severity:
WARNING

65900 (2)

Explanation:
System error. A server error occurred during an attempt to open a file for a purpose other than reading; that is, with NEW or REPLACE.

Module:
DMSOPDBK

Severity:
ERROR

65900 (3)

Explanation:
System error. A server error occurred during an attempt to open a file for a purpose other than reading, that is, with NEW, WRITE, or REPLACE.

Module:
DMSOPEN

Severity:
ERROR

66100 (1)

Explanation:
File is in DFSMS/VM migrated status and DFSMS/VM recall processing has been disabled.

Module:
DMSFILEC

Severity:
ERROR

66100 (2)

Explanation:
File is in DFSMS/VM migrated status and DFSMS/VM recall processing had been disabled.

Module:
DMSOPBLK

Severity:
ERROR

66100 (3)

Explanation:
Input file has been moved and DFSMS/VM recall processing has been disabled.

Module:
DMSOPDBK

Severity:
ERROR

66100 (4)

Explanation:
File is in DFSMS/VM migrated status and DFSMS/VM recall processing has been disabled.

Module:
DMSOPEN

Severity:
ERROR

66100 (5)

Explanation:
Input file has been migrated and DFSMS/VM recall processing has been disabled.

Module:
DMSTRUNC

Severity:
ERROR

66200

Explanation:
User selected for rollback by the SFS File Space Usage exit. The work unit will be rolled back.

Module:
Common

Severity:
ERROR

66300

Explanation:
SFS has rejected a request for a session lock on behalf of another user.

Module:
DMSCRLOC

Severity:
ERROR

66400

Explanation:
The storage group was not be enabled, because it contains an FBA minidisk that is not aligned on a 4K boundary. This restriction applies only if you have created data spaces since the file pool server started up.

Module:
DMSENASG

Severity:
ERROR

67000

Explanation:
Function specified is incorrect (currently the only valid function is RENAME).

Module:
DMSENAFS

Severity:
ERROR

69000 (1)

Explanation:
Only one BFS object in a file pool may be updated on a single work unit. The work unit may not include updates to other BFS objects in the same file pool or any SFS objects in that file pool.

Module:
Common

Severity:
ERROR

69000 (2)

Explanation:
Only 1 file space ID can be specified if the file space type is BFS.

Module:
DMSENUSR

Severity:
ERROR

69200

Explanation:
You cannot lock a BFS file SHARE.

Module:
DMSCRLOC

Severity:
ERROR

69300

Explanation:
Object is not a BFS regular file. You cannot create a lock on a BFS directory.

Module:
DMSCRLOC

Severity:
ERROR

71000

Explanation:
System error in file pool server reading a file pool catalog.

Module:
Common

Severity:
ERROR

71100

Explanation:
System error in file pool server writing to a file pool catalog.

Module:

Common

Severity:
ERROR

71200 (1)

Explanation:
The server encountered an error when it attempted to access a file.

Module:
Common

Severity:
ERROR

71200 (2)

Explanation:
The server encountered an error when it attempted to access a file.

Module:
DMSOPDBK

Severity:
ERROR

71200 (3)

Explanation:
Either SFS made an error in accessing the file, or you specified the same block number more than once.

Module:
DMSRDBLK

Severity:
ERROR

71200 (4)

Explanation:
The server encountered an error when it attempted to access a file.

Module:
DMSTRUNC

Severity:
ERROR

71200 (5)

Explanation:
Either SFS made an error in accessing the file, or you specified the same block number more than once.

Module:
DMSWRBLK

Severity:
ERROR

71300

Explanation:
System error in file pool server locking function.

Module:
Common

Severity:
ERROR

71400

Explanation:
System error in file pool server query function.

Module:
Common

Severity:
ERROR

71500

Explanation:
System error in file pool server storage management function.

Module:
Common

Severity:
ERROR

71600

Explanation:
System error in file pool server system services function.

Module:
Common

Severity:
ERROR

71700

Explanation:
System error in file pool server query function.

Module:
Common

Severity:
ERROR

71800

Explanation:
Request denied because there is a file pool server conflict with Coordinated Resource Recovery (CRR) resynchronization activity. FPERROR field FPEAUGMT contains the Recovery Token of the resynchronization activity.

Module:
Common

Severity:
ERROR

72000

Explanation:
User has attempted an Open Catalog request for any FILESPACE or DIRECTORY intent and a FILEPOOL RENAME command is currently in process.

Module:
DMSOPCAT

Severity:
ERROR

72100

Explanation:
User has attempted a Read Catalog request for a GROUP intent other than WRITE and a FILEPOOL RENAME command is currently in process on a file space in that storage group.

Module:
DMSRDCAT

Severity:
ERROR

72900

Explanation:
The requested file pool operation is not supported for the specified file pool.

Module:
Common

Severity:
ERROR

73000

Explanation:
System error. Invalid request input to file pool server.

Module:
Common

Severity:
ERROR

73100

Explanation:
System error. Conflicting count of connected communication paths between user machine and server machine.

Module:
Common

Severity:
ERROR

73200

Explanation:
System error. For a Close Catalog request for a file space opened with intent to write, an EXCLUSIVE explicit lock on the storage group was not found.

Module:
DMSCLCAT

Severity:
ERROR

73300

Explanation

System error in the file pool server's data access component. Only one user is affected. To restore service, either:

- Re-IPL CMS and reconnect to the server
- Or use the DMSPURWU CSL routine to purge the work unit IDs.

Module:
Common

Severity:
ERROR

74000

Explanation:
System error. Inconsistent file pool catalogs.

Module:
Common

Severity:
ERROR

74100

Explanation:
System error in migrating or recalling a file.

Module:
Common

Severity:
ERROR

75000 (1)

Explanation:
Service level of your machine is not compatible with the service level of the file pool server machine.

Module:
Common

Severity:
ERROR

75000 (2)

Explanation:
The service levels of the CRR recovery server and the user machine are not compatible.

Module:
DMSCHREG

Severity:

ERROR

75000 (3)

Explanation:
The service levels of the CRR recovery server and the user machine are not compatible.

Module:
DMSGETRS

Severity:
ERROR

75000 (4)

Explanation:
The service levels of the CRR recovery server and the user machine are not compatible.

Module:
DMSREG

Severity:
ERROR

76000 (1)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSCATTR

Severity:
ERROR

76000 (2)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back. Applicable only when COMMIT parameter is specified.

Module:
DMSCLBLK

Severity:
ERROR

76000 (3)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSCLCAT

Severity:
ERROR

76000 (4)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSCLDIR

Severity:
ERROR

76000 (5)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSCLOSE

Severity:
ERROR

76000 (6)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back (COMMIT option only).

Module:
DMSCRALI

Severity:
ERROR

76000 (7)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back (COMMIT parameter only).

Module:
DMSCRDIR

Severity:
ERROR

76000 (8)

Explanation:
Error in commit processing.

Module:
DMSCRFIL

Severity:
ERROR

76000 (9)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSCRLOC

Severity:
ERROR

76000 (10)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSDELOC

Severity:
ERROR

76000 (11)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSDEUSR

Severity:
ERROR

76000 (12)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSDISFS

Severity:
ERROR

76000 (13)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSDISSG

Severity:
ERROR

76000 (14)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSENAFS

Severity:
ERROR

76000 (15)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSENASG

Severity:
ERROR

76000 (16)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSENUSR

Severity:
ERROR

76000 (17)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT option was specified.

Module:
DMSERASE

Severity:
ERROR

76000 (18)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSEXIDI

Severity:
ERROR

76000 (19)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSEXIFI

Severity:
ERROR

76000 (20)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSEXIST

Severity:
ERROR

76000 (21)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT option is specified.

Module:
DMSFILEC

Severity:
ERROR

76000 (22)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT parameter is specified.

Module:
DMSGRANT

Severity:
ERROR

76000 (23)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSRELBK

Severity:
ERROR

76000 (24)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSRELOC

Severity:
ERROR

76000 (25)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSRENAM

Severity:
ERROR

76000 (26)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back. Applicable only if the COMMIT parameter was specified.

Module:
DMSREVOK

Severity:
ERROR

76000 (27)

Explanation:
System error in file pool server commit function. The current unit of work has been rolled back.

Module:
DMSWRACC

Severity:
ERROR

76002

Explanation:
I/O error encountered while reading the file pool catalog.

Module:
Common

Severity:
ERROR

76004

Explanation:
A file pool system limit has been reached.

Module:
Common

Severity:
ERROR

76010

Explanation:
The request was partially successful. An attempt to delete a file space ID in the list failed after at least one file space ID had been deleted. Use the *userid_index* field of the *wuerror* buffer to identify the failing file space ID.

Module:
DMSDEUSR

Severity:
WARNING

76050

Explanation:
An implicit rollback occurred. This code appears only as part of the work unit error information.

Module:
Common

Severity:
ERROR

76055

Explanation:
Invalid FPERROR number specified. It is less than 1 or greater than the number of FPERROR fields returned.

Module:
DMSWUERR

Severity:
ERROR

76056

Explanation:
The *wuerror* buffer contained no FPERROR information.

Module:
DMSWUERR

Severity:
ERROR

76060

Explanation:
An atomic SFS request has been rolled back. This code appears only as part of the work unit error information.

Module:
Common

Severity:
ERROR

76070 (1)

Explanation:
Changes to non-recoverable files have been lost. This code appears only as part of the work unit error information that is provided when you specify the *wuerror* parameter, or when you request SFS error data using the DMSGETSP (Get Synchronization Point Errors) or DMSGETER (Get My Errors) CSL routines.

Module:
DMSCOMM

Severity:
WARNING

76070 (2)

Explanation:
Changes to non-recoverable files have been lost. This code is returned in the *error_reascode* field of FPERROR. The FPERROR information is returned only if a data area is specified with the *wuerror* parameter.

Module:
DMSRETWU

Severity:
WARNING

76070 (3)

Explanation:
Changes to SFS nonrecoverable files have been lost. This code appears only as part of the work unit error information that is provided when you specify the *wuerror* parameter, or when you request SFS error data using the DMSGETSP (Get Synchronization Point Errors) or DMSGETER (Get My Errors) CSL routines.

Module:
DMSROLLB

Severity:
WARNING

76400

Explanation:
Exchange log names failed with the CRR recovery server.

Module:
Common

Severity:
ERROR

77150

Explanation:
The file pool limit of minidisks is reached with the addition of new minidisks.

Module:
Common

Severity:
WARNING

78001

Explanation:
The *exit_name* parameter is incorrect or is not followed by a blank.

Module:
DMSREG

Severity:
ERROR

78002

Explanation:
The *tpn* parameter is not 0–24 bytes in length.

Module:
DMSREG

Severity:
ERROR

78003 (1)

Explanation:
Incorrect *registry_token* parameter.

Module:
DMSCHREG

Severity:
ERROR

78003 (2)

Explanation:
Incorrect *registry_token* parameter.

Module:

DMSSETR

Severity:
ERROR

78003 (3)

Explanation:
Invalid *registry_token* parameter.

Module:
DMSUNREG

Severity:
ERROR

78004

Explanation:
Incorrect *action* parameter.

Module:
DMSSETR

Severity:
ERROR

78005

Explanation:
The *local_fully_qualified_luname* parameter is not 0–17 bytes in length.

Module:
DMSREG

Severity:
ERROR

78006

Explanation:
The *remote_fully_qualified_luname* parameter is not 0–17 bytes in length.

Module:
DMSREG

Severity:
ERROR

78007

Explanation:
The *mode_name* parameter is not 0–8 bytes in length.

Module:
DMSREG

Severity:
ERROR

78009

Explanation:
The keyword preceding the *pip_data* variable or *resource_recovery_tpn* variable is not either PIP or TPN.

Module:
DMSREG

Severity:
ERROR

78010

Explanation:
The PIP *pip_data* parameter is not 5–28 bytes in length.

Module:
DMSREG

Severity:
ERROR

78011

Explanation:
The TPN *resource_recovery_tpn* parameter is not 5–28 bytes in length.

Module:
DMSREG

Severity:
ERROR

78012 (1)

Explanation:
The *recovery_token* parameter is not 0–8 bytes in length.

Module:
DMSCHREG

Severity:
ERROR

78012 (2)

Explanation:
The *recovery_token* parameter is not 0–8 bytes in length.

Module:
DMSREG

Severity:
ERROR

78013

Explanation:
The *resource_component_id* parameter is not 0 or 9 bytes in length.

Module:
DMSREG

Severity:
ERROR

78014

Explanation:
The *simple_commit_flag* parameter is not 0 or 1.

Module:
DMSREG

Severity:
ERROR

78015 (1)

Explanation:
The *write_mode_flag* parameter is not 0, 1, or 2.

Module:
DMSCHREG

Severity:
ERROR

78015 (2)

Explanation:
The *write_mode_flag* parameter is not 0 or 1.

Module:
DMSREG

Severity:
ERROR

78016 (1)

Explanation:
The *single_writer_flag* parameter is not 0, 1, or 2.

Module:
DMSCHREG

Severity:
ERROR

78016 (2)

Explanation:
The *single_writer_flag* parameter is not 0 or 1.

Module:
DMSREG

Severity:
ERROR

78017 (1)

Explanation:
The *function_flags* parameter is not five characters long, or contains a character other than 0, 1, or 2.

Module:
DMSCHREG

Severity:
ERROR

78017 (2)

Explanation:
The *function_flags* parameter does not contain five character, or contains a character other than 0 or 1.

Module:
DMSREG

Severity:
ERROR

78018

Explanation:
The *session_instance_id* parameter is not 0–8 characters long.

Module:
DMSREG

Severity:
ERROR

78019

Explanation:
The *access_instance_id* parameter is not 0–8 characters long.

Module:
DMSREG

Severity:
ERROR

78020

Explanation:
Invalid synchronous option specified; must be SYNC or ASYNC.

Module:
DMSSSPTO

Severity:
ERROR

78021

Explanation:
Invalid commit decision option specified; must be BACKOUT or COMMIT.

Module:
DMSSSPTO

Severity:
ERROR

78022

Explanation:
Invalid work unit option specified; must be ALL or SINGLE.

Module:
DMSSSPTO

Severity:
ERROR

78023

Explanation:
Invalid LUWID parameter.

Module:

Common

Severity:
ERROR

78024

Explanation:
Incorrect *error_block_buffer_length* parameter.

Module:
DMSREG

Severity:
ERROR

78100

Explanation:
System error determining the node ID of the server.

Module:
Common

Severity:
ERROR

78101

Explanation:
Incorrect overwrite attribute specified (must be NOTINPLACE or INPLACE) or incorrect recoverability attribute specified (must be RECOVER or NORECOVER).

Module:
DMSPUSHA

Severity:
ERROR

78102

Explanation:
Duplicate attribute specification for a given file mode number.

Module:
DMSPUSHA

Severity:
ERROR

78104

Explanation:
No file attributes were placed on the stack.

Module:
DMSPOPA

Severity:
WARNING

78105

Explanation:
Incorrect block number passed.

Module:
DMSWRBLK

Severity:
ERROR

78106

Explanation:
The file already has the specified attributes.

Module:
DMSCATTR

Severity:
WARNING

78107 (1)

Explanation:
Recoverability attribute does not match that of the base file. For a BFS file, the NORECOVER attribute is not allowed.

Module:
DMSOPBLK

Severity:
WARNING

78107 (2)

Explanation:
Recoverability attribute does not match that of the base file. The NORECOVER attribute is not allowed for a BFS file.

Module:
DMSOPDBK

Severity:
WARNING

78107 (3)

Explanation:
Recoverability attribute does not match that of the base file. The NORECOVER attribute is not allowed for BFS.

Module:
DMSOPEN

Severity:
WARNING

78108 (1)

Explanation:
Overwrite attribute does not match that of the base file. For a BFS file, the INPLACE attribute is not allowed.

Module:
DMSOPBLK

Severity:
WARNING

78108 (2)

Explanation:
Overwrite attribute does not match that of the base file. The INPLACE attribute is not allowed for a BFS file.

Module:
DMSOPDBK

Severity:
WARNING

78108 (3)

Explanation:
Overwrite attribute does not match that of the base file. The INPLACE attribute is not allowed for BFS.

Module:
DMSOPEN

Severity:
WARNING

78110

Explanation:
ACF and recoverability and/or overwrite attributes were specified. These attributes are ignored if the base file is a file or alias.

Module:
Common

Severity:
WARNING

78111

Explanation:
An alias name was specified in the input. The base file name is returned in the fqfn field of the actual name output operand.

Module:
Common

Severity:
WARNING

78112

Explanation:
A file in DFSMS/VM migrated status was encountered during a READ CATALOG operation.

Module:
DMSRDCAT

Severity:
WARNING

79051

Explanation:
The routine identified by the *exit_name* parameter or *backout_exit_name* parameter is not loaded.

Module:
DMSREG

Severity:
ERROR

79052

Explanation:
The CSL template for the exit routine defined an incorrect number of parameters.

Module:
DMSREG

Severity:
ERROR

79053 (1)

Explanation:
The CRR recovery server is not available, so the resource registration has been changed to single writer.

Module:
DMSCHREG

Severity:
WARNING

79053 (2)

Explanation:
The CRR recovery server is not available, so the resource has been registered as a single writer.

Module:
DMSREG

Severity:
WARNING

79054 (1)

Explanation:
The CRR recovery server is needed to do a two-phase commit of the work unit, but the server is not available.

Module:
DMSCHREG

Severity:
ERROR

79054 (2)

Explanation:
The CRR recovery server is not available but is needed to do a two-phase commit of the work unit.

Module:
DMSREG

Severity:
ERROR

79055 (1)

Explanation:

Write mode was set on, but a different resource is already in write mode for the work unit with single writer set on.

Module:
DMSCHREG

Severity:
ERROR

79055 (2)

Explanation:
The *write_mode_flag* parameter was set on for this resource, but a different resource is already in write mode for the work unit with the *single_writer_flag* parameter set on.

Module:
DMSREG

Severity:
ERROR

79056 (1)

Explanation:
Write mode and single writer were set on, but a different resource is already in write mode for the work unit.

Module:
DMSCHREG

Severity:
ERROR

79056 (2)

Explanation:
The *write_mode_flag* parameter and *single_writer_flag* parameter were set on for this resource, but a different resource is already in write mode for the work unit.

Module:
DMSREG

Severity:
ERROR

79057

Explanation:
The work unit is ending.

Module:
DMSREG

Severity:
ERROR

79058 (1)

Explanation:
Change Registration failed because a sync point is in progress for the work unit.

Module:
DMSCHREG

Severity:
ERROR

79058 (2)

Explanation:
A synchronization point is already in progress for the work unit. This request was ignored.

Module:
DMSCOMM

Severity:
ERROR

79058 (3)

Explanation:
A sync point is in progress for the work unit.

Module:
DMSREG

Severity:
ERROR

79058 (4)

Explanation:
A synchronization point is already in progress for the work unit. This request was ignored.

Module:
DMSRETWU

Severity:
ERROR

79058 (5)

Explanation:
A synchronization point is already in progress for the work unit. This request was ignored.

Module:
DMSROLLB

Severity:
ERROR

79058 (6)

Explanation:
The call to DMSSETR is not allowed because the SPM is already processing a sync point.

Module:
DMSSETR

Severity:
ERROR

79059

Explanation:

An action value of backout or resource failure was given, but the work unit is already in a state that requires a backout.

Module:
DMSSETR

Severity:
WARNING

79060

Explanation:
The work unit is already associated with an LUWID..

Module:
Common

Severity:
ERROR

79061 (1)

Explanation

A resource is not in a state that can be committed. This request was ignored, the work unit was not rolled back or committed, and all protected resources were left in the same state they were prior to the call to DMSCOMM. Reasons this could occur in a protected conversation include:

- A protected conversation is not in send, defer, or synchronization point state on a commit function
- A protected conversation is in send state but did not finish sending a logical record (applies to basic conversation only) on an attempt to commit.

Reasons this could occur with SFS include:

- A catalog is open (see DMSOPCAT).
- Committing the work would exceed the user's file space limit.

Module:
DMSCOMM

Severity:
ERROR

79061 (2)

Explanation

A resource reported that the application left it in a state that does not allow a rollback, so this request was ignored and the work unit was not rolled back or committed. All protected resources are in the same state they were before the call to DMSROLLB. Use the Get Synchronization Point Errors CSL routine (DMSGETSP) to find out which protected resource caused the error. If the error occurred on a protected conversation (one allocated with a sync level of sync point), the application must deallocate the conversation. A protected conversation can cause a rollback to fail when an application:

- Has allocated a protected conversation, but its partner has not accepted
- Has sent data on a protected conversation, but its partner has not received the data
- Issued a Receive on a protected conversation with a nonzero buffer length, but has not received all of the data yet.

Module:
DMSROLLB

Severity:
ERROR

79062

Explanation:
Coupled Commit is not possible.

Module:
Common

Severity:
WARNING

79063

Explanation:
A different resource is also registered for the work unit.

Module:
Common

Severity:
ERROR

81050

Explanation:
System error. An unexpected return code was received from a call to DMSCSL fast path initialization for the *exit_name* routine or *backout_exit_name* routine.

Module:
DMSREG

Severity:
ERROR

81051 (1)

Explanation

A protocol violation may have provoked an inconsistent state among the resources.

RC=16
The COMMIT operation completed. One or more protected resources may have rolled back.

RC=20
The COMMIT operation failed. The work unit was rolled back, but one or more protected resources may have committed.

Module:
DMSCOMM

Severity:
SEVERE ERROR

81051 (2)

Explanation
A protocol violation may have provoked an inconsistent state among the resources.

RC=16
The commit or rollback operation was completed. One or more protected resources may have rolled back.

RC=20
The commit or rollback operation failed. The work unit was rolled back, but one or more protected resources may have committed.

Module:
DMSRETWU

Severity:
SEVERE ERROR

81051 (3)

Explanation:
When trying to roll back changes, one or more protected resources may have committed due to a protocol violation.

Module:
DMSROLLB

Severity:
SEVERE ERROR

81052 (1)

Explanation
Resource manager or operator intervention resulted in an inconsistent state among the resources.

RC=16
The COMMIT operation completed. One or more protected resources have rolled back.

RC=20
The COMMIT operation failed. When trying to roll back changes, one or more protected resources committed.

Module:
DMSCOMM

Severity:
SEVERE ERROR

81052 (2)

Explanation
Resource manager or operator intervention resulted in an inconsistent state among the resources.

RC=16
The commit operation was completed. Protected resources have been rolled back.

RC=20
The commit operation failed. During an attempt to roll back changes, protected resources were committed.

Module:
DMSRETWU

Severity:
SEVERE ERROR

81053 (1)

Explanation:
The work unit was rolled back; all protected resources have been restored to their prior mutually consistent state. A resource manager (such as SFS) or a conversation partner detected a problem and initiated a rollback.

Module:
DMSCOMM

Severity:
ERROR

81053 (2)

Explanation:
The work unit was returned successfully, but outstanding work was rolled back.

Module:
DMSRETWU

Severity:
ERROR

81053 (3)

Explanation:
The work unit was rolled back. All protected resources have been restored to their prior mutually consistent state. The rollback did not occur because of normal processing, but was caused by an occurrence such as a problem with one of the protected resources.

Module:
DMSROLLB

Severity:
ERROR

81054 (1)

Explanation:
Resynchronization is in progress on one or more protected resources and their consistency state is unknown. In other words, if the resynchronization is successful, all protected resources will be committed and if it is not successful, changes to all protected resources will be rolled back.

Module:
DMSCOMM

Severity:
WARNING

81054 (2)

Explanation:
The COMMIT operation failed. A resynchronization operation is in progress to finish rolling back one or more protected resources. Their consistency state is unknown. If the resynchronization is successful, all resources will be rolled back.

Module:
DMSCOMM

Severity:
ERROR

81054 (3)

Explanation:
Resynchronization is in progress on one or more protected resources and the consistency state is unknown. In other words, if the resynchronization is successful, all resources are committed.

Module:
DMSRETWU

Severity:
WARNING

81054 (4)

Explanation:
The commit operation failed. A resynchronization operation is in progress to finish rolling back one or more protected resources. Their consistency state is unknown. If the resynchronization is successful, all protected resources will be rolled back.

Module:
DMSRETWU

Severity:
ERROR

81055 (1)

Explanation:
All protected resources have been committed. However, the availability of a resource has changed. For example, an open file may have been closed.

Module:
DMSCOMM

Severity:
WARNING

81055 (2)

Explanation:
All protected resources have committed. However, your processing environment has changed or the availability of one or more resources has changed. For example, some protected conversations may have been deallocated or some open files may have been closed.

Module:
DMSRETWU

Severity:
WARNING

81055 (3)

Explanation:
The rollback was successful, but your processing environment has been changed. For example, some protected conversations may have been deallocated, or updates to nonrecoverable SFS files may have been lost. To get resource-specific error information, use the DMSGETSP (Get Synchronization Point Errors) CSL routine; or, for SFS errors, use the contents of the *wuerror* parameter.

Module:
DMSROLLB

Severity:
WARNING

81056 (1)

Explanation:
The CRR recovery server is not available. The work unit was rolled back; all protected resources have been restored to their prior mutually consistent state.

Module:
DMSCOMM

Severity:
ERROR

81056 (2)

Explanation:
The CRR recovery server is unavailable.

Module:
DMSGETRS

Severity:
ERROR

81056 (3)

Explanation:

The CRR recovery server is not available. The work unit was rolled back; all protected resources have been restored to their prior mutually consistent state.

Module:
DMSRETWU

Severity:
ERROR

81057

Explanation:
System error. The resource is not registered for the work unit.

Module:
DMSSETR

Severity:
ERROR

81058 (1)

Explanation:
System error. An addressing exception occurred while accessing data in a data space owned by a file pool server machine.

Module:
DMSOPDBK

Severity:
ERROR

81058 (2)

Explanation:
Server failure. The specified directory was placed in a data space, but your authorization to that data space was revoked because of server failure. To regain your authorization to the data space, simply reexecute the routine when the server is available. (The server internally handles data space authorizations based on your authority to the directory.)

Module:
DMSOPDIR

Severity:
ERROR

81058 (3)

Explanation:
System error. An addressing exception occurred while accessing data in a data space owned by a file pool server machine.

Module:
DMSOPEN

Severity:
ERROR

81059

Explanation:
The work unit was rolled back due to an application error. Two or more protected conversation partners initiated a commit at the same time.

Module:
DMSCOMM

Severity:
ERROR

85004 (1)

Explanation:
The specified data space name matches the name of a previously created data space. The data space was not created. The previously created data space was not created using DMSSPCC.

Module:
DMSSPCC

Severity:
ERROR

85004 (2)

Explanation:
The specified address space identification token (ASIT) does not identify a data space that was created by your virtual machine. This could occur if a data space is not created, or if it is created using DMSSPCC, but then is deleted using a CP macro before calling DMSSPCD.

Module:
DMSSPCD

Severity:
ERROR

85004 (3)

Explanation:
The specified address space identification token (ASIT) does not identify a data space that was created by your virtual machine. This could occur if a data space is not created, or if it is created using DMSSPCC, but then is deleted using a CP macro before calling DMSSPCD.

Module:
DMSSPCI

Severity:
ERROR

85004 (4)

Explanation:
The specified address space identification token (ASIT) does not identify a data space that was created by your virtual machine. This could occur if a data space is not created, or if it is created using DMSSPCC, but then is deleted using a CP macro before calling DMSSPCD.

Module:
DMSSPCP

Severity:
ERROR

85004 (5)

Explanation
Protection exception (Code X'0004')

- A part of the target area is in CP-page-protected storage.
- The PSW key is not valid for accessing either the source or target storage.

Module:
DMSSPCPY

Severity:
ERROR

85004 (6)

Explanation:
You do not own or have access to the address. No information was returned.

Module:
DMSSPCQ

Severity:
ERROR

85004 (7)

Explanation:
The address space identification token specified in the *asit* parameter does not identify an address space owned by the issuing virtual machine. No authorization was allowed.

Module:
DMSSPCR

Severity:
ERROR

85004 (8)

Explanation:
The address space identification token specified for the *asit* parameter does not identify a valid address space (one owned by the caller or one to which the caller has been permitted access). The ALET is not returned and addressability is not established to the address space.

Module:
DMSSPLA

Severity:
ERROR

85004 (9)

Explanation:
The ALE designated by the access list entry token (ALET) was already in an unused state. The state of the ALE is unchanged.

Module:
DMSSPLR

Severity:
ERROR

85005 (1)

Explanation
Addressing exception (Code X'0005')

- The source area extends into storage locations not available in the address space.
- The target area extends into storage locations not available in your primary address space.

Module:
DMSSPCPY

Severity:
ERROR

85005 (2)

Explanation
Addressing exception (Code X'0005')

- An attempt to release a page outside its defined storage.

Module:
DMSSPCRP

Severity:
ERROR

85006

Explanation
Specification exception (Code X'0006')

- An attempt was made to release page 0 of the primary address space.

Module:
DMSSPCRP

Severity:
ERROR

85008 (1)

Explanation:
Creating the data space would cause the maximum number of allowed address spaces for your virtual machine to be exceeded. The data space was not created.

Module:

DMSSPCC

Severity:
ERROR

85008 (2)

Explanation:
Error in the parameter specification.

Module:
DMSSPCPY

Severity:
ERROR

85008 (3)

Explanation:
There are no available entries in the access list which can be used for the current request. The ALET is not returned and access is not allowed to the address space.

Module:
DMSSPLA

Severity:
ERROR

85012 (1)

Explanation:
Creating the data space would cause the maximum total data space size for your virtual machine to be exceeded. The data space was not created.

Module:
DMSSPCC

Severity:
ERROR

85012 (2)

Explanation:
The ALET specified in the *alet* parameter is not valid. The state of the ALET is unchanged.

Module:
DMSSPLR

Severity:
ERROR

85016 (1)

Explanation:
The specified data space name contains incorrect characters. The data space was not created.

Module:
DMSSPCC

Severity:
ERROR

85016 (2)

Explanation:
The specified address space name contains incorrect characters. No information was returned.

Module:
DMSSPCQ

Severity:
ERROR

85024 (1)

Explanation:
All virtual machines identified by the *user* or *user_list* parameter have already been authorized to access the address space. There was no change in the virtual machines' authorization.

Module:
DMSSPCP

Severity:
WARNING

85024 (2)

Explanation:
All virtual machines that were previously permitted and represented by CMS structures have already been authorized to access the address space. There was no change in the virtual machines' authorization.

Module:
DMSSPCR

Severity:
WARNING

85028 (1)

Explanation:
All of the virtual machines identified by the *user* or *user_list* parameter are not either currently logged on or disconnected. No authorization was allowed.

Module:
DMSSPCP

Severity:
ERROR

85028 (2)

Explanation:
All of the virtual machines that were previously permitted and represented by CMS structures are either not currently logged on or are disconnected. No authorization was allowed.

Module:
DMSSPCR

Severity:
ERROR

85032

Explanation:

The issuer's XCONFIG ADDRSPACE directory statement did not have the SHARE option specified. See the *z/VM: CP Planning and Administration* for more information on directory entries.

Module:
DMSSPCP

Severity:
ERROR

85040 (1)

Explanation
ALET specification exception (Code X'0028')

- The *alet* is invalid.

Module:
DMSSPCPY

Severity:
ERROR

85040 (2)

Explanation
ALET specification exception (Code X'0028')

- The *alet* is incorrect.

Module:
DMSSPCRP

Severity:
ERROR

85041 (1)

Explanation
ALEN translation exception (Code X'0029')

- The *alet* designates an access list entry in an unused state.

Module:
DMSSPCPY

Severity:
ERROR

85041 (2)

Explanation
ALEN translation exception (Code X'0029')

- The *alet* designates an ALE (access list entry) in an unused state.

Module:
DMSSPCRP

Severity:
ERROR

85100 (1)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCC

Severity:
ERROR

85100 (2)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCD

Severity:
ERROR

85100 (3)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCI

Severity:
ERROR

85100 (4)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCP

Severity:
ERROR

85100 (5)

Explanation:
This support is not available or this service was called from an XC virtual machine.

Module:
DMSSPCPY

Severity:
ERROR

85100 (6)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCQ

Severity:
ERROR

85100 (7)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCR

Severity:
ERROR

85100 (8)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPCRP

Severity:
ERROR

85100 (9)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPLA

Severity:
ERROR

85100 (10)

Explanation:
This support is not provided by the processor on which VM is running.

Module:
DMSSPLR

Severity:
ERROR

85102 (1)

Explanation
Incorrect parameter or options specified. This could be caused by:

- An incorrect value
- Two options having a duplicate value
- One option's value conflicting with another's.

Module:
DMSSPCC

Severity:
ERROR

85102 (2)

Explanation
Incorrect parameter or options specified. This could be provoked by:

- An incorrect value
- Two options having a duplicate value
- One option's value conflicting with another's.

Module:
DMSSPCI

Severity:
ERROR

85102 (3)

Explanation
Incorrect parameter or options specified. The error could be:

- An incorrect value
- Two options having a duplicate value
- One option’s value conflicting with another’s.

Module:
DMSSPCP

Severity:
ERROR

85102 (4)

Explanation
Incorrect parameter or options specified. The error could be:

- An incorrect value
- Two options having a duplicate value
- One option's value conflicting with another's.

Module:
DMSSPLA

Severity:
ERROR

85103

Explanation:
The specified address space does not have the CMS SHARE attribute specified to allow access from another virtual machine. No authorization was allowed.

Module:
DMSSPCP

Severity:
ERROR

85104 (1)

Explanation:
Illegal request for a CMS data space service. For example, the requesting virtual machine tried to create a data space with a name that already exists due to a previous call to DMSSPCC or the specified data space name begins with the preassigned CMS prefix, DMS.

Module:
DMSSPCC

Severity:
ERROR

85104 (2)

Explanation:
Illegal request for a data space service. For example, DMSSPCC was not called to create the data space specified for deletion on this call.

Module:
DMSSPCD

Severity:
ERROR

85104 (3)

Explanation:
Illegal request for a CMS address space service. For example, DMSSPCC was not called to create the data space specified for isolation on this call, or the ASIT is incorrect.

Module:
DMSSPCI

Severity:
ERROR

85104 (4)

Explanation:
Illegal request for a address space service. For example, DMSSPCC was not called to create the data space specified on this permit request.

Module:
DMSSPCP

Severity:
ERROR

85104 (5)

Explanation:
Illegal request for a CMS address space service. For example, PURGE was specified on the preceding DMSSPCI routine and therefore, there are no users with permission to access the address space. They must have their access restored with the DMSSPCP routine. This reason code can also mean that DMSSPCC was not called to create the address space designated on this call.

Module:
DMSSPCR

Severity:
ERROR

85104 (6)

Explanation:
Illegal request for a CMS address space service. For example, the requesting virtual machine does not own the address space in which pages are to be released.

Module:
DMSSPCRP

Severity:
ERROR

85105 (1)

Explanation:
A call sequence warning occurred. DMSSPCI was called for an address space that was not previously specified on a Permit (DMSSPCP) or Restore (DMSSPCR). Either DMSSPCI was called immediately after the address space was created or it was called immediately after a prior DMSSPCI call. The address space was already in a private state, so this call had no effect.

Module:
DMSSPCI

Severity:
WARNING

85105 (2)

Explanation:
A call sequence warning occurred. This call was not preceded by a call to DMSSPCI. Any users that currently had access still have it; any users that were logged off have been removed from the CMS-maintained list of users with access to the address space.

Module:
DMSSPCR

Severity:
WARNING

85105 (3)

Explanation:
A call sequence error occurred. Addressability has not been established to the address space. Either DMSSPLA was not called prior to this call, or DMSSPLR was called to remove the addressability.

Module:
DMSSPCRP

Severity:
ERROR

85106

Explanation:
Incorrect value specified for the *user_count* parameter. No user IDs were processed.

Module:
DMSSPCP

Severity:
ERROR

85107 (1)

Explanation:
The virtual machine list identified by *user_list* has been processed, but there was at least one user ID that was already authorized or that was not currently logged on.

Module:
DMSSPCP

Severity:
WARNING

85107 (2)

Explanation:
The previously permitted virtual machines now have their address space access restored, but there was at least one invalid (for example, logged off) entry detected in the CMS-maintained permit list. Any user found to be logged off is removed from the permit list.

Module:
DMSSPCR

Severity:
WARNING

85108 (1)

Explanation:
An ESTAE-related problem occurred that could jeopardize the ability to report an interrupt code in the event of a program check.

Module:
DMSSPCPY

Severity:
ERROR

85108 (2)

Explanation:
An ESTAE-related problem occurred that could jeopardize the ability to report an interrupt code in the event of a program check.

Module:
DMSSPCRP

Severity:
ERROR

85310 (1)

Explanation

Addressing capability exception (Code X'0136')

- The *alet* designates an ALE that is in revoked state.

Module:
DMSSPCPY

Severity:
ERROR

85310 (2)

Explanation

Addressing capability exception (Code X'0136')

- The *alet* designates an ALE (access list entry) in the revoked state.

Module:
DMSSPCRP

Severity:
ERROR

86300

Explanation:
The CP product information could not be determined from the licensed program bit map contained in the CMS control blocks, possibly because of corrupted storage.

Module:
DMSQEFL

Severity:
ERROR

86400 (1)

Explanation:
The CMS level information could not be determined, possibly because of corrupted storage.

Module:
DMSQEFL

Severity:
ERROR

86400 (2)

Explanation:
File pool server level information could not be determined, possibly because of corrupted storage.

Module:
DMSQSFSL

Severity:
ERROR

Reason Codes 90101-90472 generated by the VMLIB CSL routines

Explanation, originating module, and severity is documented for reason codes 90101-90472, which are generated by VMLIB CSL routines.

90101 (1)

Explanation:
Read was successful, but the output buffer was too small to hold all of the requested data. The data is truncated to the buffer size.

Module:
DMSRDDBK

Severity:
WARNING

90101 (2)

Explanation:
Read request was successful, but the buffer was too small to hold all of the requested data. The data is truncated to the buffer size.

Module:
DMSREAD

Severity:
WARNING

90101 (3)

Explanation:
Data truncated. *return_line* was not big enough to hold the returned value.

Module:
DMSTCD

Severity:
WARNING

90102 (1)

Explanation:
Read was successful, but the output buffer was too small to hold all of the requested data, and your file space warning threshold was reached or exceeded. The data is truncated to the buffer size.

Module:
DMSRDDBK

Severity:
WARNING

90102 (2)

Explanation:
Read request was successful, but the buffer was too small to hold all of the requested data, and your file space warning threshold was reached or exceeded. The data is truncated to the buffer size.

Module:
DMSREAD

Severity:
WARNING

90103 (1)

Explanation:
No blocks read. End of file was reached, or the position parameter specified a record number greater than the number of blocks in the file.

Module:
DMSRDDBK

Severity:
WARNING

90103 (2)

Explanation:
No records read. End of file was reached, or the position parameter specified a record number greater than the number of records in the file.

Module:
DMSREAD

Severity:
WARNING

90105 (1)

Explanation:
Incorrect record format.

Module:
DMSOPDBK

Severity:
ERROR

90105 (2)

Explanation:
Incorrect record format.

Module:
DMSOPEN

Severity:
ERROR

90105 (3)

Explanation:
Invalid record format.

Module:
DMSREAD

Severity:
ERROR

90105 (4)

Explanation:

Incorrect record format.

Module:
DMSWRITE

Severity:
ERROR

90106 (1)

Explanation:
The input number of records was invalid. This must be a positive integer value.

Module:
DMSCLDBK

Severity:
ERROR

90106 (2)

Explanation:
Number of records to read is not greater than zero.

Module:
DMSREAD

Severity:
ERROR

90106 (3)

Explanation:
Number of records to write is not greater than zero.

Module:
DMSWRITE

Severity:
ERROR

90107 (1)

Explanation:
Number of records to read is not exactly one for a file containing variable length records.

Module:
DMSREAD

Severity:
ERROR

90107 (2)

Explanation:
Number of records to write is not exactly one for a file containing variable-length records.

Module:
DMSWRITE

Severity:
ERROR

90108 (1)

Explanation:
Size of output buffer is not greater than zero.

Module:
DMSRDDBK

Severity:
ERROR

90108 (2)

Explanation:
Size of buffer is not greater than zero.

Module:
DMSREAD

Severity:
ERROR

90108 (3)

Explanation:
Size of input or output buffer is not greater than zero or an attempt to write a null record to a file having variable-length records.

Module:
DMSWRDBK

Severity:
ERROR

90108 (4)

Explanation:
Size of input buffer is not greater than zero, or you have attempted to write a null record to a file containing variable-length records.

Module:
DMSWRITE

Severity:
ERROR

90109

Explanation:
Specified data length is not evenly divisible by the number of records to be written to a file containing fixed-length records.

Module:
DMSWRITE

Severity:
ERROR

90110

Explanation:
Size of input buffer is greater than 65535 for a file containing variable length records.

Module:
DMSWRITE

Severity:
ERROR

90111 (1)

Explanation:
Invalid buffer address.

Module:
DMSRDDBK

Severity:
ERROR

90111 (2)

Explanation:
Invalid buffer address.

Module:
DMSREAD

Severity:
ERROR

90111 (3)

Explanation:
Invalid buffer address.

Module:
DMSWRDBK

Severity:
ERROR

90111 (4)

Explanation:
Incorrect buffer address.

Module:
DMSWRITE

Severity:
ERROR

90112 (1)

Explanation:
Position specifies a negative block number.

Module:
DMSRDDBK

Severity:
ERROR

90112 (2)

Explanation:
Position specifies a negative record number.

Module:
DMSREAD

Severity:
ERROR

90112 (3)

Explanation:
Position specifies a negative block number.

Module:
DMSWRDBK

Severity:
ERROR

90112 (4)

Explanation:
Position specifies a negative record number.

Module:
DMSWRITE

Severity:
ERROR

90113 (1)

Explanation:
Position plus the number of blocks to read exceeds 2^{31} - 1, which is the file system capacity.

Module:
DMSRDDBK

Severity:
ERROR

90113 (2)

Explanation:
Position plus the number of records to read exceeds 2^{31}-1, which is the file system capacity.

Module:
DMSREAD

Severity:
ERROR

90113 (3)

Explanation:
Position plus the number of blocks to write exceeds 2^{31} - 1, the file system capacity.

Module:
DMSWRDBK

Severity:
ERROR

90113 (4)

Explanation:
Position plus the number of records to write exceeds 2^{31} - 1, the file system capacity.

Module:
DMSWRITE

Severity:
ERROR

90114 (1)

Explanation:

In a variable format file, the position of the write pointer is neither at zero nor at the start of the next block.

Module:
DMSWRDBK

Severity:
ERROR

90114 (2)

Explanation:
Position specifies a record number that is more than one greater than the current number of records in a file containing variable length records.

Module:
DMSWRITE

Severity:
ERROR

90115

Explanation

File system capacity exceeded; a write operation attempted to put more than 2^{31} - 1 times blocksize bytes of data in a file, requiring a logical block number greater than 2^{31} - 1. This can occur for files having either fixed-length records or variable-length records:

- For a file with fixed-length records, it occurs when the product of the record number of the last record to be written and the logical record length is greater than 2^{31} - 1 times the block size.
- For a file with variable-length records, it occurs when the sum of the byte offset of a variable-length record and the length of that record is greater than 2^{31} - 1 times the block size.

Module:
DMSWRITE

Severity:
ERROR

90116

Explanation:
Invalid DMSRCMFL operation (value not in the range 0-2 for an SFS file).

Module:
Common

Severity:
ERROR

90117

Explanation:
The variable length record read is invalid. The length is either zero or outside of the range (1 to logical record length).

Module:
DMSREAD

Severity:
ERROR

90118 (1)

Explanation:
Invalid value in the *error_block_counter* parameter. The caller changed the value in the *error_block_counter* parameter or a synchronization point occurred since the call to DMSGETER that created the counter.

Module:
DMSGETER

Severity:
ERROR

90118 (2)

Explanation:
Invalid value in the *error_block_counter* parameter. The caller changed the value in the *error_block_counter* parameter or a synchronization point occurred since the call to DMSGETSP that created it.

Module:
DMSGETSP

Severity:
ERROR

90119

Explanation:
File is not new.

Module:
Common

Severity:
ERROR

90120

Explanation:
You have attempted to alter the record length of a file containing fixed length records.

Module:
DMSWRITE

Severity:
ERROR

90121

Explanation:
You have attempted to replace an existing variable-length record with one of a different length.

Module:
DMSWRITE

Severity:
ERROR

90128

Explanation:
Unable to obtain space on the system stack for a file system module's dynamic storage.

Module:
Common

Severity:
ERROR

90129 (1)

Explanation:
There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available. Applicable only when COMMIT parameter is specified.

Module:
DMSCLBLK

Severity:
ERROR

90129 (2)

Explanation:
There are already 2^{31}-1 blocks in the file, therefore a new logical block is not available.

Module:
DMSCLDBK

Severity:
ERROR

90129 (3)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSCLDIR

Severity:
ERROR

90129 (4)

Explanation:
There are already 2^{31} - 1 blocks in the file, therefore a new logical block is not available.

Module:
DMSCLOSE

Severity:
ERROR

90129 (5)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSCRALI

Severity:
ERROR

90129 (6)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSCRDIR

Severity:
ERROR

90129 (7)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in the file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSERASE

Severity:
ERROR

90129 (8)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSEXIDI

Severity:
ERROR

90129 (9)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSEXIFI

Severity:
ERROR

90129 (10)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit; therefore a new logical block is not available.

Module:
DMSEXIST

Severity:
ERROR

90129 (11)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSFILEC

Severity:
ERROR

90129 (12)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSGRANT

Severity:
ERROR

90129 (13)

Explanation:
There are already 2^{31} - 1 blocks in the file.

Module:
DMSRDDBK

Severity:
ERROR

90129 (14)

Explanation:
There are already 2^{31}-1 blocks in the file.

Module:
DMSREAD

Severity:
ERROR

90129 (15)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in the file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSRENAM

Severity:
ERROR

90129 (16)

Explanation:
COMMIT was specified. There are already 2^{31}-1 blocks in a file to which you have written in the same work unit, therefore a new logical block is not available.

Module:
DMSREVOK

Severity:
ERROR

90129 (17)

Explanation:
File system capacity exceeded: the design limit does not permit more than 2^{31} - 1 blocks to be allocated to the file.

Module:
DMSWRDBK

Severity:
ERROR

90129 (18)

Explanation:
File system capacity exceeded: the design limit does not permit more than 2^{31} - 1 blocks to be allocated to the file.

Module:
DMSWRITE

Severity:
ERROR

90130 (1)

Explanation:
Insufficient free virtual storage available for file system control blocks.

Module:
Common

Severity:
ERROR

90130 (2)

Explanation:
There is not enough free virtual storage available for file system control blocks.

Module:
DMSCLOSE

Severity:
ERROR

90131 (1)

Explanation:
Insufficient minidisk space available.

Module:
DMSCLDBK

Severity:
ERROR

90131 (2)

Explanation:
Insufficient minidisk space available.

Module:
DMSFILEC

Severity:
ERROR

90131 (3)

Explanation:
Insufficient minidisk space available.

Module:
DMSTRUNC

Severity:
ERROR

90131 (4)

Explanation:
Insufficient minidisk space available.

Module:
DMSWRDBK

Severity:
ERROR

90131 (5)

Explanation:
Insufficient minidisk space for a write operation to a minidisk file.

Module:
DMSWRITE

Severity:
ERROR

90132

Explanation:
Permanent I/O error during a read operation to an EDF file.

Module:
Common

Severity:
ERROR

90133

Explanation:
Permanent I/O error during a write operation to an EDF file.

Module:
Common

Severity:
ERROR

90140

Explanation:
Invalid file mode letter.

Module:
Common

Severity:
ERROR

90144

Explanation:
The FORCE option was specified and some or all file updates are not permanent.

Module:
DMSWRITE

Severity:
WARNING

90145

Explanation:
The FORCE option was specified, some or all file updates are not permanent and file space warning threshold reached or exceeded.

Module:
DMSWRITE

Severity:
WARNING

90146 (1)

Explanation:
Nonzero bytes follow end of data.

Module:
DMSFILEC

Severity:
ERROR

90146 (2)

Explanation:
Nonzero bytes follow end of data.

Module:
DMSWRDBK

Severity:
ERROR

90160

Explanation:
Insufficient buffer size. Must be a multiple of 4096 (4K).

Module:
DMSFILEC

Severity:
ERROR

90180

Explanation:
Buffer size not equal to the number of blocks multiplied by the block size.

Module:
DMSWRDBK

Severity:
ERROR

90200 (1)

Explanation:
Specified file mode number does not match the file mode number of the specified file or external object.

Module:
DMSEXIFI

Severity:
WARNING

90200 (2)

Explanation:
Specified file mode number does not match the file mode number of the specified file or external object.

Module:
DMSEXIST

Severity:
WARNING

90210 (1)

Explanation:
Extraneous characters in file pool ID specification.

Module:
DMSDEUSR

Severity:
ERROR

90210 (2)

Explanation:
Extraneous characters in input parameter.

Module:
DMSDISFS

Severity:
ERROR

90210 (3)

Explanation:
Extraneous characters in one of the following parameters: SHARE|EXCLUSIVE, or DETACH|NODETACH.

Module:
DMSDISSG

Severity:
ERROR

90210 (4)

Explanation:
Extraneous characters in an input parameter.

Module:
DMSENAFS

Severity:
ERROR

90210 (5)

Explanation:
Extraneous characters in file pool ID specification.

Module:
DMSENUSR

Severity:
ERROR

90210 (6)

Explanation:
Extraneous characters in input parameter.

Module:
DMSGETER

Severity:
ERROR

90210 (7)

Explanation:
Extraneous characters present in an input parameter.

Module:
DMSOPBLK

Severity:
ERROR

90210 (8)

Explanation:
Extraneous characters in input parameter.

Module:
DMSOPCAT

Severity:
ERROR

90210 (9)

Explanation:
Extraneous characters in an input parameter.

Module:
DMSQFPDS

Severity:
ERROR

90210 (10)

Explanation:
Extraneous characters in input parameter.

Module:
DMSQLIMA

Severity:
ERROR

90210 (11)

Explanation:
Extraneous characters in input parameter.

Module:
DMSQLIMU

Severity:
ERROR

90210 (12)

Explanation:
Extraneous characters in file pool ID.

Module:
DMSQSFSL

Severity:
ERROR

90215 (1)

Explanation:
Incorrect buffer length.

Module:
DMSQFPDS

Severity:
ERROR

90215 (2)

Explanation:
Incorrect buffer length.

Module:
DMSQLIMA

Severity:
ERROR

90215 (3)

Explanation:
Incorrect buffer length.

Module:
DMSQUSG

Severity:
ERROR

90215 (4)

Explanation:
Invalid buffer length.

Module:
DMSRDCAT

Severity:
ERROR

90215 (5)

Explanation:
Invalid buffer length.

Module:
DMSWRCAT

Severity:
ERROR

90215 (6)

Explanation:
Invalid buffer length specified for input *wuerror* buffer.

Module:
DMSWUERR

Severity:
ERROR

90215 (7)

Explanation:
The length of the buffer is not greater than or equal to zero.

Module:
StackRead—DMSSTKR

Severity:
ERROR

90216 (1)

Explanation:
The specified request ID does not match any active request.

Module:
DMSCHECK

Severity:
ERROR

90216 (2)

Explanation:
Invalid *requestid* parameter. There is no such request ID active at this time.

Module:

DMSMARK

Severity:
ERROR

90217 (1)

Explanation:
Incorrect record number specified: it is less than or equal to zero or it is greater than the number of records being returned in the buffer.

Module:
DMSQFPDD

Severity:
ERROR

90217 (2)

Explanation:
The requested element is less than or equal to zero, or the requested element is greater than the number of records being returned in the buffer.

Module:
DMSUSGD

Severity:
ERROR

90220 (1)

Explanation:
The specified file does not exist or you are not authorized to erase it.

Module:
DMSERASE

Severity:
ERROR

90220 (2)

Explanation:
The specified file, external object, or directory does not exist or you are not authorized to it.

Module:
DMSEXIST

Severity:
ERROR

90220 (3)

Explanation:
The specified file does not exist in the specified directory, or no files exist that match the specified wildcard pattern.

Module:
DMSOPDIR

Severity:
ERROR

90220 (4)

Explanation:
TCPIP DATA file not found.

Module:
DMSTCD

Severity:
WARNING

90221

Explanation:
Array element specified is not valid. It is less than 1 or greater than the returned number of users enrolled.

Module:
DMSQLIMD

Severity:
ERROR

90222

Explanation:
The asynchronous requests have not been completed.

Module:
DMSCHECK

Severity:
WARNING

90223

Explanation:
Asynchronous calls may not be made from REXX.

Module:
Common

Severity:
ERROR

90230 (1)

Explanation:
The specified directory does not exist or you are not authorized to it.

Module:
DMSERASE

Severity:
ERROR

90230 (2)

Explanation:
The specified directory does not exist or you are not authorized to it.

Module:
DMSEXIST

Severity:
ERROR

90230 (3)

Explanation:

Directory does not exist or you are not authorized to grant authority to it.

Module:
DMSGRANT

Severity:
ERROR

90230 (4)

Explanation:
The specified directory does not exist or you are not authorized to it.

Module:
DMSOPDIR

Severity:
ERROR

90230 (5)

Explanation:
Specified directory does not exist or you are not authorized to revoke authorities granted on it.

Module:
DMSREVOK

Severity:
ERROR

90245 (1)

Explanation:
Directory was opened for a different intent. It must be opened with intent of SEARCHALL.

Module:
DMSGETDA

Severity:
ERROR

90245 (2)

Explanation:
Directory was opened for a different intent. It must be opened with intent of DIR.

Module:
DMSGETDD

Severity:
ERROR

90245 (3)

Explanation:
Directory was opened for a different intent. It must be opened with intent of FILE.

Module:
DMSGETDF

Severity:
ERROR

90245 (4)

Explanation:
Directory was opened for a different intent. It must be opened with intent of LOCK.

Module:
DMSGETDK

Severity:
ERROR

90245 (5)

Explanation:
Directory was opened for a different intent. It must be opened with intent of ALIAS.

Module:
DMSGETDL

Severity:
ERROR

90245 (6)

Explanation:
Directory was opened for a different intent. It must be opened with intent of SEARCHAUTH.

Module:
DMSGETDS

Severity:
ERROR

90245 (7)

Explanation:
Directory was opened for a different intent. It must be opened with intent of AUTH.

Module:
DMSGETDT

Severity:
ERROR

90245 (8)

Explanation:
Directory was opened for a different intent. It must be opened with intent of FILEEXT.

Module:
DMSGETDX

Severity:
ERROR

90250 (1)

Explanation:
The file name and file type (or *namedef*) are required but were not specified.

Module:
DMSCATTR

Severity:
ERROR

90250 (2)

Explanation:
The file name and file type (or *namedef*) are required but were not specified.

Module:
DMSCRALI

Severity:
ERROR

90250 (3)

Explanation:
File name and file type (or *namedef*) are required, but were not specified.

Module:
DMSCRFIL

Severity:
ERROR

90250 (4)

Explanation:
File name and file type (or *namedef*) are required, but were not specified.

Module:
DMSCROB

Severity:
ERROR

90250 (5)

Explanation:
The file name and file type (or *namedef*) are required but were not specified.

Module:
DMSEXIFI

Severity:
ERROR

90250 (6)

Explanation:
The file name and file type (or namedef) are required but were not specified.

Module:
DMSFILEC

Severity:
ERROR

90250 (7)

Explanation:
The file name and file type (or *namedef*) are required but were not specified.

Module:
DMSOPBLK

Severity:
ERROR

90250 (8)

Explanation:
The file name and file type (or *namedef*) are required but were not specified.

Module:
DMSOPDBK

Severity:
ERROR

90250 (9)

Explanation:
File ID must be specified for Open Directory for SEARCHALL, SEARCHAUTH, ALIAS, FILE, or FILEEXT.

Module:
DMSOPDIR

Severity:
ERROR

90250 (10)

Explanation:
The file name and file type (or *namedef*) are required but were not specified.

Module:
DMSOPEN

Severity:
ERROR

90250 (11)

Explanation:
File name and file type (or *namedef*) are required, but were not specified.

Module:
DMSQOBJ

Severity:
ERROR

90255 (1)

Explanation:
A file name, file type, or file mode number may not be specified in a *dirname* parameter.

Module:
DMSEXIDI

Severity:
ERROR

90255 (2)

Explanation:

File ID cannot be specified for Open Directory for DIR.

Module:
DMSOPDIR

Severity:
ERROR

90260 (1)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDA

Severity:
ERROR

90260 (2)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDD

Severity:
ERROR

90260 (3)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDF

Severity:
ERROR

90260 (4)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDI

Severity:
ERROR

90260 (5)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDK

Severity:
ERROR

90260 (6)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDL

Severity:
ERROR

90260 (7)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDS

Severity:
ERROR

90260 (8)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDT

Severity:
ERROR

90260 (9)

Explanation:
The directory has been closed. The directory was erased, or your authority to it was revoked.

Module:
DMSGETDX

Severity:
ERROR

90270 (1)

Explanation:
Output buffer was too small to contain all of the requested output. The output has been truncated to the buffer length.

Module:
DMSCLCAT

Severity:
WARNING

90270 (2)

Explanation:
Output buffer is too small to contain the requested fixed length output.

Module:
DMSEXIST

Severity:

ERROR

90270 (3)

Explanation:
Output buffer is too small to contain one record.

Module:
DMSGETDI

Severity:
ERROR

90270 (4)

Explanation:
Output buffer was too small to contain all of the requested output. The output has been truncated to the buffer length.

Module:
DMSOPCAT

Severity:
WARNING

90270 (5)

Explanation:
Output buffer is too small. Some records have been discarded.

Module:
DMSQFPDS

Severity:
WARNING

90270 (6)

Explanation:
Output buffer is too small. Data has been truncated.

Module:
DMSQLIMA

Severity:
WARNING

90270 (7)

Explanation:
Warning: input buffer is too small.

Module:
DMSQUSG

Severity:
WARNING

90270 (8)

Explanation:
Buffer is not large enough to hold at least one block. No data read.

Module:
DMSRDDBK

Severity:
ERROR

90270 (9)

Explanation:
The returned directory name is longer than the length specified in *length2*.

Module:
DMSVALDT

Severity:
ERROR

90271 (1)

Explanation:
All available information would not fit in the *error_data_buffer* buffer. As much data as would fit in the buffer has been placed there. Check the *actual_error_data_length* parameter to get the actual length of the usable data, make the buffer equal to the returned length, set the *error_block_counter* parameter to 0, and call DMSGETER again.

Module:
DMSGETER

Severity:
WARNING

90271 (2)

Explanation:
Warning, all available information would not fit in the *error_data_buffer* buffer. As much data as would fit in the buffer has been placed there. Check the *actual_error_data_length* parameter to get the actual length of the usable data.

Module:
DMSGETSP

Severity:
WARNING

90271 (3)

Explanation:
Full error block. Some error data was lost.

Module:
DMSPCAER

Severity:
WARNING

90275 (1)

Explanation:
No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDA

Severity:
ERROR

90275 (2)

Explanation:
No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDD

Severity:
ERROR

90275 (3)

Explanation:
No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDF

Severity:
ERROR

90275 (4)

Explanation:
No data found for this Get Directory request. For Open Directory LOCK or ALIAS, no locks or aliases were found. For other types of Open Directory, your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDI

Severity:
ERROR

90275 (5)

Explanation:
No data found for this Get Directory request. No locks were found or your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDK

Severity:
ERROR

90275 (6)

Explanation:
No data found for this Get Directory request. No aliases were found, or your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDL

Severity:
ERROR

90275 (7)

Explanation:
No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDS

Severity:
ERROR

90275 (8)

Explanation:
No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDT

Severity:
ERROR

90275 (9)

Explanation:
No data found for this Get Directory request. Your previous Get Directory request issued warning 44040 to indicate that all of the requested data has been returned.

Module:
DMSGETDX

Severity:
ERROR

90276

Explanation:
No response data available for the previous Open Catalog or Close Catalog.

Module:
DMSCPYBF

Severity:
ERROR

90277

Explanation:
No matching registered resource was found for the input parameters you specified.

Module:
DMSGETER

Severity:
ERROR

90278 (1)

Explanation:
No error blocks were found. Either there were no synchronization point warnings or errors for this work unit, or the participant that received the warnings or errors did not create any error blocks.

Module:
DMSGETER

Severity:
WARNING

90278 (2)

Explanation:
No error blocks were found. This means that there were no synchronization point errors for this work unit, or that the protected resource did not create any error blocks.

Module:
DMSGETSP

Severity:
WARNING

90278 (3)

Explanation:
No error data was available (*actual_error_data_length* =4).

Module:
DMSPCAER

Severity:
WARNING

90280

Explanation:
More data was requested than will fit in the output buffer. There is more data to be returned.

Module:
DMSGETDI

Severity:
WARNING

90283

Explanation:
Data length specified exceeds the size of the input buffer.

Module:
DMSWRITE

Severity:
ERROR

90290 (1)

Explanation:
Incorrect number of records specified. Number of records must be equal to or greater than one.

Module:
DMSGETDI

Severity:
ERROR

90290 (2)

Explanation:
Incorrect number of records: Value must be between zero and the number of records currently in the file; or value was zero and ALLOWEMPTY was not specified for an SFS file.

Module:
DMSTRUNC

Severity:
ERROR

90300 (1)

Explanation:
Incorrect parameter in CSL parameter list.

Module:
DMSCATTR

Severity:
ERROR

90300 (2)

Explanation:
Illegal parameter specified. The COMMIT or NOCOMMIT parameter was missing or specified incorrectly, or SHORTDATE, FULLDATE, or ISODATE was specified incorrectly, or extraneous parameters were found.

Module:
DMSCLDBK

Severity:
ERROR

90300 (3)

Explanation:
Illegal parameter specified. The COMMIT/NOCOMMIT parameter was missing or specified incorrectly, or extraneous parameters were found.

Module:
DMSCLDIR

Severity:
ERROR

90300 (4)

Explanation:
Illegal parameter specified. The COMMIT or NOCOMMIT parameter was missing or specified incorrectly, or SHORTDATE, FULLDATE, or ISODATE was specified incorrectly, or extraneous parameters were found.

Module:
DMSCLOSE

Severity:
ERROR

90300 (5)

Explanation:
Invalid parameter in CSL parameter list. Must be COMMIT or NOCOMMIT.

Module:
DMSCRALI

Severity:
ERROR

90300 (6)

Explanation:
Invalid input parameter in CSL parameter list. Valid parameters are COMMIT, NOCOMMIT, FILECONTROL, DIRCONTROL, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCRDIR

Severity:
ERROR

90300 (7)

Explanation:
Invalid lock description parameter - (SHARE, UPDATE, or EXCLUSIVE, and SESSION or LASTING).

Module:
DMSCRLOC

Severity:
ERROR

90300 (8)

Explanation:
Incorrect parameter in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCROB

Severity:
ERROR

90300 (9)

Explanation:
Invalid input parameter in CSL parameter list. Specified file space ID is longer than 8 characters. Or, keyword specified preceding *length4* parameter is not valid.

Module:
DMSDEUSR

Severity:
ERROR

90300 (10)

Explanation:
Invalid parameter in CSL parameter list: either 1) the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL; 2) a file mode number was included with the file mode; or 3) a file name or file type was specified as part of the directory name.

Module:
DMSDIRAT

Severity:
ERROR

90300 (11)

Explanation:
Invalid input parameter in CSL parameter list. For example, a specified SFS or BFS file space ID is longer than 8 characters, begins with a plus (+) or minus (-), or contains a colon (:) or period (.), or a specified BFS file space ID contains a slash (/) or null character (X'00').

Module:
DMSENUSR

Severity:
ERROR

90300 (12)

Explanation:
Invalid input parameter in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSEXIDI

Severity:
ERROR

90300 (13)

Explanation:
Invalid input parameter in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSEXIFI

Severity:
ERROR

90300 (14)

Explanation:
Invalid parameter specified. The COMMIT/NOCOMMIT parameter must contain "COMMIT" or "NOCOMMIT", possibly followed by "UNIQUEID".

Module:
DMSEXIST

Severity:
ERROR

90300 (15)

Explanation:
Invalid input parameter.

Module:
DMSGETER

Severity:
ERROR

90300 (16)

Explanation:
Invalid input parameter in CSL parameter list. The specified transaction tag length is invalid.

Module:
DMSGETWU

Severity:
ERROR

90300 (17)

Explanation:
Parameter is not valid: FILEATTR format parameter specified incorrectly.

Module:
DMSOPCAT

Severity:
ERROR

90300 (18)

Explanation:
Incorrect intent parameter. Valid intents are NEW, READ, or REPLACE.

Module:
DMSOPDBK

Severity:
ERROR

90300 (19)

Explanation:
Invalid parameter specified. Type of Open Directory is invalid, extraneous keywords are present, or FORCERO/FORCERW specified without intent FILE.

Module:
DMSOPDIR

Severity:
ERROR

90300 (20)

Explanation:
Incorrect intent specified. Valid intents are NEW, READ, WRITE, or REPLACE.

Module:
DMSOPEN

Severity:
ERROR

90300 (21)

Explanation:
Invalid input parameter in CSL parameter list. The parameter was not ALL.

Module:
DMSPOPWU

Severity:
ERROR

90300 (22)

Explanation:
Invalid input parameter in CSL parameter list. The only valid input parameter is FORCE.

Module:
DMSPURWU

Severity:
ERROR

90300 (23)

Explanation:
Incorrect options in CSL parameter list.

Module:
DMSPUSHA

Severity:
ERROR

90300 (24)

Explanation:
The object type parameter (GROUP, FILESPACE *filespaceid*, or ALL) or its length is incorrect.

Module:
DMSQFPDS

Severity:
ERROR

90300 (25)

Explanation:
Invalid input parameter in CSL parameter list. Specified file space ID is longer than 8 characters.

Module:
DMSQLIMU

Severity:
ERROR

90300 (26)

Explanation:
Invalid parameter in CSL parameter list.

Module:
DMSQOBJ

Severity:
ERROR

90300 (27)

Explanation:
Incorrect parameter in CSL parameter list. File mode number erroneously specified, or file name and file type is specified in the target *dirname*.

Module:
DMSRELOC

Severity:
ERROR

90300 (28)

Explanation:
Incorrect parameter in CSL parameter list. It must be COMMIT or NOCOMMIT. COMMIT is required for subdirectories, and this reason code is returned if NOCOMMIT is specified.

Module:
DMSRENAM

Severity:
ERROR

90300 (29)

Explanation:
Invalid input parameter in CSL parameter list. The specified transaction tag length is invalid.

Module:
DMSSETAG

Severity:
ERROR

90300 (30)

Explanation:
Illegal parameter specified. The COMMIT or NOCOMMIT parameter was missing or specified incorrectly, or the ALLOWEMPTY, SHORTDATE, FULLDATE, or ISODATE parameters were specified incorrectly, or extraneous parameters were found.

Module:
DMSTRUNC

Severity:
ERROR

90302

Explanation:
Incorrect input parameter in CSL parameter list. The parameter was not ALL or the parameter length was incorrect.

Module:
DMSPOPA

Severity:
ERROR

90303

Explanation:
A DRA field was specified and OPEN intent was not RECALL. DRA is ignored.

Module:
Common

Severity:
ERROR

90304

Explanation:
A DRA field was specified with an invalid length. It must be 0 or 8.

Module:
Common

Severity:
ERROR

90305

Explanation:
FILES option is invalid when you are erasing a file.

Module:
DMSERASE

Severity:
ERROR

90306

Explanation:
The specified *lrecl* did not match the logical record length of the existing fixed-length format file.

Module:
DMSOPEN

Severity:
WARNING

90307 (1)

Explanation:
Invalid *lrecl* specified. Must be a positive integer.

Module:

DMSCLDBK

Severity:
ERROR

90307 (2)

Explanation:
Incorrect *lrecl* specified.

Module:
DMSCRFIL

Severity:
ERROR

90307 (3)

Explanation:
lrecl must not be negative.

Module:
DMSOPDBK

Severity:
ERROR

90307 (4)

Explanation:
lrecl must not be negative.

Module:
DMSOPEN

Severity:
ERROR

90308

Explanation:
Specified option is invalid for directory.

Module:
DMSERASE

Severity:
ERROR

90310 (1)

Explanation:
Incorrect option in CSL parameter list. Valid parameters are COMMIT or NOCOMMIT, and SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCLBLK

Severity:
ERROR

90310 (2)

Explanation:
Incorrect option in CSL parameter list. Valid options are COMMIT or NOCOMMIT, F or V, NORECOVER or RECOVER, INPLACE or NOTINPLACE, and SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCRFIL

Severity:
ERROR

90310 (3)

Explanation:
Invalid option in CSL parameter list. Specified user ID is greater than 8 characters in length.

Module:
DMSDELOC

Severity:
ERROR

90310 (4)

Explanation:
Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSENUSR

Severity:
ERROR

90310 (5)

Explanation:
Incorrect option in CSL parameter list: COMMIT, NOCOMMIT, DATAONLY, ENTIRE, or FILES not specified correctly.

Module:
DMSERASE

Severity:
ERROR

90310 (6)

Explanation:
Incorrect option in CSL parameter list. Valid options are COMMIT, NOCOMMIT, REPLACE, OLDDATE, NEWDATE, OLDDATEREF, NEWDATEREF, OLDCREATE, and NEWCREATE.

Module:
DMSFILEC

Severity:
ERROR

90310 (7)

Explanation:
Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSGETDA

Severity:
ERROR

90310 (8)

Explanation:
Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSGETDF

Severity:
ERROR

90310 (9)

Explanation:
Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSGETDS

Severity:
ERROR

90310 (10)

Explanation:
Incorrect option in CSL parameter list. Valid options are SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSGETDX

Severity:
ERROR

90310 (11)

Explanation:
Invalid option in CSL parameter list. For example, a specified user ID is longer than 8 characters, begins with a plus (+) or a minus (–), or contains a colon (:) or a period (.).

Module:
DMSGRANT

Severity:
ERROR

90310 (12)

Explanation:
Incorrect option in CSL parameter list.
Valid options are: NEW, READ, WRITE, REPLACE or CREATEMIG; NEWDATEREF or OLDDATEREF; RECOVER or NORECOVER; INPLACE or NOTINPLACE; ALLOWEMPTY; NORECALL; RESOLVE; and SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSOPBLK

Severity:
ERROR

90310 (13)

Explanation:
Incorrect option specified. Valid options are CACHE or NOCACHE, F or V, OLDDATEREF or NEWDATEREF, RECOVER or NORECOVER, INPLACE or NOTINPLACE, ALLOWEMPTY, SHORTDATE or FULLDATE or ISODATE.

Module:
DMSOPDBK

Severity:
ERROR

90310 (14)

Explanation:
Incorrect keyword specified. Valid keywords are NEW or READ or REPLACE or WRITE, CACHE or NOCACHE, F or V, OLDDATEREF or NEWDATEREF, RECOVER or NORECOVER, INPLACE or NOTINPLACE, OPENRECOVER, ALLOWEMPTY, SHORTDATE or FULLDATE or ISODATE.

Module:
DMSOPEN

Severity:
ERROR

90310 (15)

Explanation:
Invalid option in CSL parameter list - must be REFRESH or NOREFRESH.

Module:
DMSREAD

Severity:
ERROR

90310 (16)

Explanation:
Incorrect option in CSL parameter list.

Module:
DMSREVOK

Severity:
ERROR

90310 (17)

Explanation:
Incorrect option in CSL parameter list - must be FORCE or NOFORCE.

Module:
DMSWRITE

Severity:
ERROR

90315 (1)

Explanation:
Missing option in CSL parameter list. Specified user ID is all blanks.

Module:
DMSDELOC

Severity:
ERROR

90315 (2)

Explanation:
Missing parameter in CSL parameter list: COMMIT or NOCOMMIT not specified.

Module:
DMSERASE

Severity:
ERROR

90315 (3)

Explanation:
Missing option in CSL parameter list.

Module:
DMSGRANT

Severity:
ERROR

90315 (4)

Explanation:
The open intent (NEW, READ, or REPLACE) was not specified.

Module:
DMSOPDBK

Severity:
ERROR

90315 (5)

Explanation:
Missing option in CSL parameter list.

Module:
DMSREVOK

Severity:
ERROR

90320 (1)

Explanation:
Conflicting parameters in CSL parameter list.

Module:
DMSCATTR

Severity:
ERROR

90320 (2)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSCLBLK

Severity:
ERROR

90320 (3)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSCLDBK

Severity:
ERROR

90320 (4)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSCLOSE

Severity:
ERROR

90320 (5)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive options.

Module:
DMSCRALI

Severity:
ERROR

90320 (6)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, FILECONTROL and DIRCONTROL, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSCRDIR

Severity:
ERROR

90320 (7)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, F and V, if specified, are mutually exclusive parameters, RECOVER and NORECOVER are mutually exclusive parameters, INPLACE and NOTINPLACE are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSCRFIL

Severity:
ERROR

90320 (8)

Explanation

Conflicting parameters in CSL parameter list. The following options are mutually exclusive:

UPDATE vs. EXCLUSIVE vs. SHARE
SESSION vs. LASTING

Module:
DMSCRLOC

Severity:
ERROR

90320 (9)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSCROB

Severity:
ERROR

90320 (10)

Explanation:
Conflicting options in CSL parameter list: the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL.

Module:
DMSDIRAT

Severity:
ERROR

90320 (11)

Explanation:
Conflicting options. DETACH was specified and the lock type was not EXCLUSIVE.

Module:
DMSDISSG

Severity:
ERROR

90320 (12)

Explanation:
Specifying both *function* and *userid* parameters is not allowed.

Module:
DMSENAFS

Severity:
ERROR

90320 (13)

Explanation:
Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSENUSR

Severity:
ERROR

90320 (14)

Explanation:
Conflicting options in CSL parameter list: COMMIT and NOCOMMIT, or DATAONLY and ENTIRE, were specified.

Module:
DMSERASE

Severity:
ERROR

90320 (15)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSEXIDI

Severity:
ERROR

90320 (16)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSEXIFI

Severity:
ERROR

90320 (17)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT, or OLDDATE and NEWDATE, or OLDDATEREF and NEWDATEREF, or OLDCREATE and NEWCREATE were both specified.

Module:
DMSFILEC

Severity:

ERROR

90320 (18)

Explanation:
Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSGETDA

Severity:
ERROR

90320 (19)

Explanation:
Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSGETDF

Severity:
ERROR

90320 (20)

Explanation:
Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSGETDS

Severity:
ERROR

90320 (21)

Explanation:
Conflicting options in CSL parameter list. SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSGETDX

Severity:
ERROR

90320 (22)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSGRANT

Severity:
ERROR

90320 (23)

Explanation:
Conflicting options in CSL parameter list. NEW, READ, WRITE, REPLACE, and CREATEMIG are mutually exclusive parameters, NEWDATEREF and OLDDATEREF, if specified, are mutually exclusive parameters, RECOVER and NORECOVER, if specified, are mutually exclusive parameters, INPLACE and NOTINPLACE, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSOPBLK

Severity:
ERROR

90320 (24)

Explanation:
Conflicting options in CSL parameter list. NEW, READ, and REPLACE are mutually exclusive parameters, CACHE and NOCACHE, if specified, are mutually exclusive parameters, F and V, if specified, are mutually exclusive parameters, NEWDATEREF and OLDDATEREF, if specified, are mutually exclusive parameters, RECOVER and NORECOVER, if specified, are mutually exclusive parameters, INPLACE and NOTINPLACE, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSOPDBK

Severity:
ERROR

90320 (25)

Explanation:
Conflicting options in CSL parameter list. NEW, READ, REPLACE, and WRITE are mutually exclusive parameters, CACHE and NOCACHE, if specified, are mutually exclusive parameters, F and V, if specified, are mutually exclusive parameters, NEWDATEREF and OLDDATEREF, if specified, are mutually exclusive parameters, RECOVER and NORECOVER, if specified, are mutually exclusive parameters, INPLACE and NOTINPLACE, if specified, are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSOPEN

Severity:
ERROR

90320 (26)

Explanation:

Invalid *error_block* or *actual_error_data_length* parameter.

Module:
DMSPCAER

Severity:
ERROR

90320 (27)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSPUSHA

Severity:
ERROR

90320 (28)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSREAD

Severity:
ERROR

90320 (29)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSREVOK

Severity:
ERROR

90320 (30)

Explanation:
Conflicting options in CSL parameter list. COMMIT and NOCOMMIT are mutually exclusive parameters, and SHORTDATE, FULLDATE, and ISODATE, if specified, are mutually exclusive parameters.

Module:
DMSTRUNC

Severity:
ERROR

90320 (31)

Explanation:
Conflicting options in CSL parameter list.

Module:
DMSWRITE

Severity:
ERROR

90330 (1)

Explanation:
Duplicate parameters in CSL parameter list.

Module:
DMSCATTR

Severity:
ERROR

90330 (2)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSCLBLK

Severity:
ERROR

90330 (3)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSCLOSE

Severity:
ERROR

90330 (4)

Explanation:
Duplicate options in CSL parameter list. COMMIT or NOCOMMIT or UNRESOLVED was specified more than once.

Module:
DMSCRALI

Severity:
ERROR

90330 (5)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, FILECONTROL, DIRCONTROL, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCRDIR

Severity:
ERROR

90330 (6)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, RECOVER, NORECOVER, INPLACE, NOTINPLACE, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCRFIL

Severity:
ERROR

90330 (7)

Explanation:
Duplicate parameters in CSL parameter list. UPDATE, EXCLUSIVE, SHARE, SESSION, LASTING was specified more than once.

Module:
DMSCRLOC

Severity:
ERROR

90330 (8)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSCROB

Severity:
ERROR

90330 (9)

Explanation:
Duplicate options in CSL parameter list: the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL.

Module:
DMSDIRAT

Severity:
ERROR

90330 (10)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSENUSR

Severity:
ERROR

90330 (11)

Explanation:
Duplicate parameter in CSL parameter list: COMMIT, NOCOMMIT, DATAONLY, ENTIRE, or FILES already specified.

Module:
DMSERASE

Severity:
ERROR

90330 (12)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSEXIDI

Severity:
ERROR

90330 (13)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSEXIFI

Severity:
ERROR

90330 (14)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSFILEC

Severity:
ERROR

90330 (15)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.

Module:
DMSGETDA

Severity:
ERROR

90330 (16)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.

Module:
DMSGETDF

Severity:
ERROR

90330 (17)

Explanation:

Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.

Module:
DMSGETDS

Severity:
ERROR

90330 (18)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: SHORTDATE, FULLDATE, and ISODATE.

Module:
DMSGETDX

Severity:
ERROR

90330 (19)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSGRANT

Severity:
ERROR

90330 (20)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: NEW, READ, WRITE, REPLACE, CREATEMIG, NEWDATEREF, OLDDATEREF, RECOVER, NORECOVER, INPLACE, NOTINPLACE, ALLOWEMPTY, NORECALL, RESOLVE, SHORTDATE, FULLDATE, or ISODATE.

Module:
DMSOPBLK

Severity:
ERROR

90330 (21)

Explanation:
Parameter not valid.

Module:
DMSOPDBK

Severity:
WARNING

90330 (22)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: NEW, READ, REPLACE, CACHE, NOCACHE, F, V, NEWDATEREF, OLDDATEREF, RECOVER, NORECOVER, INPLACE, NOINPLACE, ALLOWEMPTY, SHORTDATE, FULLDATE, and ISODATE.

Module:
DMSOPDBK

Severity:
ERROR

90330 (23)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: NEW, READ, REPLACE, WRITE, CACHE, NOCACHE, F, V, NEWDATEREF, OLDDATEREF, RECOVER, NORECOVER, INPLACE, NOTINPLACE, OPENRECOVER, ALLOWEMPTY, SHORTDATE, FULLDATE, and ISODATE.

Module:
DMSOPEN

Severity:
ERROR

90330 (24)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSPUSHA

Severity:
ERROR

90330 (25)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSREAD

Severity:
ERROR

90330 (26)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSREVOK

Severity:
ERROR

90330 (27)

Explanation:
Duplicate options in CSL parameter list. One or more of the following parameters were specified more than once: COMMIT, NOCOMMIT, ALLOWEMPTY, SHORTDATE, FULLDATE, and ISODATE.

Module:

DMSTRUNC

Severity:
ERROR

90330 (28)

Explanation:
Duplicate options in CSL parameter list.

Module:
DMSWRITE

Severity:
ERROR

90350 (1)

Explanation:
Incorrect number of blank-delimited tokens in the file ID or dirname parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSCATTR

Severity:
ERROR

90350 (2)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSCRALI

Severity:
ERROR

90350 (3)

Explanation:
Incorrect number of blank-delimited tokens in the dirname parameter.

Module:
DMSCRDIR

Severity:
ERROR

90350 (4)

Explanation:
Incorrect number of blank-delimited tokens. There must be more than one token, because *fileid* and *dirname* are required.

Module:
DMSCRFIL

Severity:
ERROR

90350 (5)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSCRLOC

Severity:
ERROR

90350 (6)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least 1 and not more than 4 tokens in the string.

Module:
DMSCROB

Severity:
ERROR

90350 (7)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSDELOC

Severity:
ERROR

90350 (8)

Explanation:
Incorrect number of blank-delimited tokens in the directory name parameter.

Module:
DMSDIRAT

Severity:
ERROR

90350 (9)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than three tokens in the string.

Module:
DMSERASE

Severity:
ERROR

90350 (10)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and no more than four tokens in the string.

Module:
DMSEXIDI

Severity:
ERROR

90350 (11)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSEXIFI

Severity:
ERROR

90350 (12)

Explanation:
Incorrect number of blank-delimited tokens in the third parameter, which identifies the object to be checked. There must be at least one and not more than four tokens in the string.

Module:
DMSEXIST

Severity:
ERROR

90350 (13)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and no more than four tokens in the string.

Module:
DMSFILEC

Severity:
ERROR

90350 (14)

Explanation:
Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least one and not more than three tokens in the string.

Module:
DMSGRANT

Severity:
ERROR

90350 (15)

Explanation:
Incorrect number of blank-delimited tokens in the file ID or directory name parameter. There must be at least 1 and not more than 4 tokens in the string.

Module:
DMSOPBLK

Severity:
ERROR

90350 (16)

Explanation:
Incorrect number of blank-delimited tokens in the *dirname*, *bfsid*, or *namedef* parameter.

Module:
DMSOPCAT

Severity:
ERROR

90350 (17)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least two and not more than four tokens in the string.

Module:
DMSOPDBK

Severity:
ERROR

90350 (18)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSOPDIR

Severity:
ERROR

90350 (19)

Explanation:
Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSOPEN

Severity:
ERROR

90350 (20)

Explanation:
Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least 2 and not more than 4 tokens in the string.

Module:
DMSQOBJ

Severity:
ERROR

90350 (21)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSRELOC

Severity:
ERROR

90350 (22)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than four tokens in the string.

Module:
DMSRENAM

Severity:
ERROR

90350 (23)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least one and not more than three tokens in the string.

Module:
DMSREVOK

Severity:
ERROR

90350 (24)

Explanation:
Incorrect number of blank-delimited tokens in the *fileid* or *dirname* parameter. There must be at least two and not more than three tokens in the string.

Module:
DMSTRUNC

Severity:
ERROR

90350 (25)

Explanation:
Incorrect number of blank-delimited tokens in the *fn_ft* or *dirname* parameter. There must be at least one and not more than three tokens in the string.

Module:
DMSUDATA

Severity:
ERROR

90350 (26)

Explanation:
The compound parameter for the file ID is incorrect. There must be two, three, or four blank-delimited tokens in the string. Namedefs are treated as single strings.

Module:
DMSVALDT

Severity:
ERROR

90351

Explanation:
DATAONLY option was specified and file is already empty.

Module:
DMSERASE

Severity:
WARNING

90380 (1)

Explanation:
Missing parameter in CSL parameter list. Recoverability and overwrite attributes must be specified.

Module:
DMSCATTR

Severity:
ERROR

90380 (2)

Explanation:
Missing parameter in CSL parameter list. COMMIT or NOCOMMIT must be specified.

Module:
DMSCRALI

Severity:
ERROR

90380 (3)

Explanation:
Missing parameter in CSL parameter list. COMMIT or NOCOMMIT is required.

Module:
DMSCRDIR

Severity:
ERROR

90380 (4)

Explanation:
Required parameter omitted in CSL parameter list.

Module:
DMSCRFIL

Severity:
ERROR

90380 (5)

Explanation

Missing parameter in CSL parameter list. One of each of the following sets of options must be specified:

- UPDATE or EXCLUSIVE or SHARE
- SESSION or LASTING

Module:
DMSCRLOC

Severity:
ERROR

90380 (6)

Explanation:
Required parameter omitted in CSL parameter list.

Module:
DMSCROB

Severity:
ERROR

90380 (7)

Explanation:
Missing parameter in CSL parameter list: the directory attribute was not DIRCONTROL, DIRCONTROL FORCE, or FILECONTROL.

Module:
DMSDIRAT

Severity:
ERROR

90380 (8)

Explanation:
Missing parameter in CSL parameter list.

Module:
DMSFILEC

Severity:
ERROR

90380 (9)

Explanation:
Missing parameter in CSL parameter list.

Module:
DMSOPBLK

Severity:
ERROR

90380 (10)

Explanation:
Missing parameter in CSL parameter list.

Module:
DMSQOBJ

Severity:
ERROR

90400

Explanation:
Both FN and FT are needed to represent a file.

Module:
Common

Severity:
ERROR

90405

Explanation:
Specified object is not an external object.

Module:
DMSQOBJ

Severity:
ERROR

90410 (1)

Explanation:
Incorrect parameter length specified.

Module:
DMSCATTR

Severity:
ERROR

90410 (2)

Explanation:
Invalid length specified for *userdata*, file ID, or keyword (COMMIT, NOCOMMIT, UNRESOLVED) parameters.

Module:
DMSCRALI

Severity:
ERROR

90410 (3)

Explanation:
Invalid length specified for the *userdata*, directory ID, or COMMIT/NOCOMMIT parameters.

Module:
DMSCRDIR

Severity:
ERROR

90410 (4)

Explanation:
Incorrect length specified for a character variable.

Module:
DMSCRFIL

Severity:

ERROR

90410 (5)

Explanation:
Invalid length specified for the *fileid*, *dirname*, *userdata*, or lock description (UPDATE, EXCLUSIVE, or SHARE, and SESSION or LASTING) parameter.

Module:
DMSCRLOC

Severity:
ERROR

90410 (6)

Explanation:
Incorrect length specified for one of the character parameters.

Module:
DMSCROB

Severity:
ERROR

90410 (7)

Explanation:
Invalid length specified for *userdata*, file ID, or directory name.

Module:
DMSDELOC

Severity:
ERROR

90410 (8)

Explanation:
Invalid length specified for directory attribute (that is, for DIRCONTROL, FILECONTROL, or DIRCONTROL FORCE).

Module:
DMSDIRAT

Severity:
ERROR

90410 (9)

Explanation:
Incorrect parameter length specified. The name in *function* has a length greater than 8.

Module:
DMSENAFS

Severity:
ERROR

90410 (10)

Explanation:
Invalid length specified for *userdata*, COMMIT, NOCOMMIT, DATAONLY, ENTIRE, or FILES parameter.

Module:
DMSERASE

Severity:
ERROR

90410 (11)

Explanation:
Invalid length specified for one of the character variables.

Module:
DMSEXIDI

Severity:
ERROR

90410 (12)

Explanation:
Invalid length specified for one of the character variables.

Module:
DMSEXIFI

Severity:
ERROR

90410 (13)

Explanation:
Incorrect length specified for one of the character variables.

Module:
DMSEXIST

Severity:
ERROR

90410 (14)

Explanation:
Incorrect length specified for one of the character parameters.

Module:
DMSFILEC

Severity:
ERROR

90410 (15)

Explanation:
Invalid parameter length specified (*sectoken* can be 0-64, and *transaction_tag* can be 0-80 characters).

Module:
DMSGETWU

Severity:
ERROR

90410 (16)

Explanation:

Invalid parameter length specified.

Module:
DMSGRANT

Severity:
ERROR

90410 (17)

Explanation:
Incorrect length specified for one of the character variables.

Module:
DMSOPBLK

Severity:
ERROR

90410 (18)

Explanation:
Incorrect length specified for one of the character variables.

Module:
DMSOPDBK

Severity:
ERROR

90410 (19)

Explanation:
Invalid length specified for one of the character variables.

Module:
DMSOPDIR

Severity:
ERROR

90410 (20)

Explanation:
Incorrect length specified for one of the character variables.

Module:
DMSOPEN

Severity:
ERROR

90410 (21)

Explanation:
Invalid length specified for one of the character parameters.

Module:
DMSQOBJ

Severity:
ERROR

90410 (22)

Explanation:
Incorrect length specified for one of the character variables.

Module:
DMSRELOC

Severity:
ERROR

90410 (23)

Explanation:
Invalid length specified for one of the character variables.

Module:
DMSRENAM

Severity:
ERROR

90410 (24)

Explanation:
Incorrect parameter length specified.

Module:
DMSREVOK

Severity:
ERROR

90410 (25)

Explanation:
Incorrect length specified for one of the character variables.

Module:
DMSTRUNC

Severity:
ERROR

90410 (26)

Explanation:
Incorrect length specified for one of the character parameters.

Module:
DMSUDATA

Severity:
ERROR

90410 (27)

Explanation:
Invalid parameter length specified.

Module:
DMSVALDT

Severity:
ERROR

90415 (1)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCATTR

Severity:
ERROR

90415 (2)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCLBLK

Severity:
ERROR

90415 (3)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCLCAT

Severity:
ERROR

90415 (4)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCLDBK

Severity:
ERROR

90415 (5)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCLDIR

Severity:
ERROR

90415 (6)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero *wuerror* length must be at least 12 bytes.

Module:
DMSCLOSE

Severity:
ERROR

90415 (7)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCOMM

Severity:
ERROR

90415 (8)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCRALI

Severity:
ERROR

90415 (9)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCRDIR

Severity:
ERROR

90415 (10)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCRFIL

Severity:
ERROR

90415 (11)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCRLOC

Severity:
ERROR

90415 (12)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSCROB

Severity:
ERROR

90415 (13)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSDELOC

Severity:
ERROR

90415 (14)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSDEUSR

Severity:
ERROR

90415 (15)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSDIRAT

Severity:
ERROR

90415 (16)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSDISFS

Severity:
ERROR

90415 (17)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSDISSG

Severity:
ERROR

90415 (18)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSENAFS

Severity:
ERROR

90415 (19)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSENASG

Severity:
ERROR

90415 (20)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSENUSR

Severity:
ERROR

90415 (21)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSERASE

Severity:
ERROR

90415 (22)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSEXIDI

Severity:
ERROR

90415 (23)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSEXIFI

Severity:
ERROR

90415 (24)

Explanation:

Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSEXIST

Severity:
ERROR

90415 (25)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSFILEC

Severity:
ERROR

90415 (26)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDA

Severity:
ERROR

90415 (27)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDD

Severity:
ERROR

90415 (28)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDF

Severity:
ERROR

90415 (29)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDI

Severity:
ERROR

90415 (30)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDK

Severity:
ERROR

90415 (31)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDL

Severity:
ERROR

90415 (32)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDS

Severity:
ERROR

90415 (33)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDT

Severity:
ERROR

90415 (34)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGETDX

Severity:
ERROR

90415 (35)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSGRANT

Severity:
ERROR

90415 (36)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSOPBLK

Severity:
ERROR

90415 (37)

Explanation:
Length specified for the *wuerror* parameter is not valid. A nonzero length must be at least 12 bytes.

Module:
DMSOPCAT

Severity:
ERROR

90415 (38)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSOPDBK

Severity:
ERROR

90415 (39)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSOPDIR

Severity:
ERROR

90415 (40)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSOPEN

Severity:
ERROR

90415 (41)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSPURWU

Severity:
ERROR

90415 (42)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSQCONN

Severity:
ERROR

90415 (43)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSQFPDS

Severity:
ERROR

90415 (44)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSQLIMU

Severity:
ERROR

90415 (45)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSQOBJ

Severity:
ERROR

90415 (46)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSQSFSL

Severity:
ERROR

90415 (47)

Explanation:

Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSQUSG

Severity:
ERROR

90415 (48)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRDBLK

Severity:
ERROR

90415 (49)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRDCAT

Severity:
ERROR

90415 (50)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRDDBK

Severity:
ERROR

90415 (51)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSREAD

Severity:
ERROR

90415 (52)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRELBK

Severity:
ERROR

90415 (53)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRELOC

Severity:
ERROR

90415 (54)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRENAM

Severity:
ERROR

90415 (55)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSRETWU

Severity:
ERROR

90415 (56)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSREVOK

Severity:
ERROR

90415 (57)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSROLLB

Severity:
ERROR

90415 (58)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSTRUNC

Severity:
ERROR

90415 (59)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSWRACC

Severity:
ERROR

90415 (60)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSWRBLK

Severity:
ERROR

90415 (61)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSWRCAT

Severity:
ERROR

90415 (62)

Explanation:
Invalid length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSWRDBK

Severity:
ERROR

90415 (63)

Explanation:
Incorrect length specified for the *wuerror* parameter. A nonzero length must be at least 12 bytes.

Module:
DMSWRITE

Severity:
ERROR

90420 (1)

Explanation:
The file name in the file ID parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSCATTR

Severity:
ERROR

90420 (2)

Explanation:
The file name is longer than 8 characters or contains an invalid character.

Module:
DMSCRALI

Severity:
ERROR

90420 (3)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than eight characters or contains an incorrect character.

Module:
DMSCRFIL

Severity:
ERROR

90420 (4)

Explanation:
The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.

Module:
DMSCRLOC

Severity:
ERROR

90420 (5)

Explanation:
The file name is incorrect. The file name is longer than eight characters or contains an incorrect character.

Module:
DMSCROB

Severity:
ERROR

90420 (6)

Explanation:
The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.

Module:
DMSDELOC

Severity:
ERROR

90420 (7)

Explanation:
The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.

Module:
DMSERASE

Severity:
ERROR

90420 (8)

Explanation:
The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.

Module:
DMSEXIFI

Severity:
ERROR

90420 (9)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSEXIST

Severity:
ERROR

90420 (10)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSFILEC

Severity:
ERROR

90420 (11)

Explanation:
The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.

Module:
DMSGRANT

Severity:
ERROR

90420 (12)

Explanation:
The file name in the file ID parameter is incorrect. The file name is longer than eight characters or contains an incorrect character.

Module:
DMSOPBLK

Severity:
ERROR

90420 (13)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSOPDBK

Severity:
ERROR

90420 (14)

Explanation:
The file name in the *fileid* parameter is invalid. The file name is longer than 8 characters or contains an invalid character.

Module:
DMSOPDIR

Severity:
ERROR

90420 (15)

Explanation:
The file name in the *fn_ft* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSOPEN

Severity:
ERROR

90420 (16)

Explanation:
The file name in the *fn_ft* parameter is invalid. The file name is longer than eight characters or contains an invalid character.

Module:
DMSQOBJ

Severity:
ERROR

90420 (17)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSRELOC

Severity:
ERROR

90420 (18)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSRENAM

Severity:
ERROR

90420 (19)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSREVOK

Severity:
ERROR

90420 (20)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSTRUNC

Severity:
ERROR

90420 (21)

Explanation:
The file name in the *fileid* parameter is incorrect. The file name is longer than 8 characters or contains an incorrect character.

Module:
DMSUDATA

Severity:
ERROR

90420 (22)

Explanation:
The file name is incorrect. The file name is longer than eight characters or contains an incorrect character.

Module:
DMSVALDT

Severity:
ERROR

90430 (1)

Explanation:
The file type in the file ID parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSCATTR

Severity:
ERROR

90430 (2)

Explanation:
The file type is longer than 8 characters or contains an invalid character.

Module:
DMSCRALI

Severity:
ERROR

90430 (3)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than eight characters or contains an incorrect character.

Module:
DMSCRFIL

Severity:
ERROR

90430 (4)

Explanation:
The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.

Module:
DMSCRLOC

Severity:
ERROR

90430 (5)

Explanation:
The file type is incorrect. The file type is longer than eight characters or contains an incorrect character.

Module:
DMSCROB

Severity:
ERROR

90430 (6)

Explanation:

The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.

Module:
DMSDELOC

Severity:
ERROR

90430 (7)

Explanation:
The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.

Module:
DMSERASE

Severity:
ERROR

90430 (8)

Explanation:
The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.

Module:
DMSEXIFI

Severity:
ERROR

90430 (9)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSEXIST

Severity:
ERROR

90430 (10)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSFILEC

Severity:
ERROR

90430 (11)

Explanation:
The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.

Module:
DMSGRANT

Severity:
ERROR

90430 (12)

Explanation:
The file type in the file ID parameter is incorrect. The file type is longer than eight characters or contains an incorrect character.

Module:
DMSOPBLK

Severity:
ERROR

90430 (13)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSOPDBK

Severity:
ERROR

90430 (14)

Explanation:
The file type in the *fileid* parameter is invalid. The file type is longer than 8 characters or contains an invalid character.

Module:
DMSOPDIR

Severity:
ERROR

90430 (15)

Explanation:
The file type in the *fn_ft* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSOPEN

Severity:
ERROR

90430 (16)

Explanation:
The file type in the *fn_ft* parameter is invalid. The file type is longer than eight characters or contains an invalid character.

Module:
DMSQOBJ

Severity:
ERROR

90430 (17)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSRELOC

Severity:
ERROR

90430 (18)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSRENAM

Severity:
ERROR

90430 (19)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSREVOK

Severity:
ERROR

90430 (20)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSTRUNC

Severity:
ERROR

90430 (21)

Explanation:
The file type in the *fileid* parameter is incorrect. The file type is longer than 8 characters or contains an incorrect character.

Module:
DMSUDATA

Severity:
ERROR

90430 (22)

Explanation:
The file type is incorrect. The file type is longer than eight characters or contains an invalid character.

Module:
DMSVALDT

Severity:
ERROR

90440 (1)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSCRFIL

Severity:
ERROR

90440 (2)

Explanation:
The specified file mode is invalid. It must be an alphabetic character.

Module:
DMSCRLOC

Severity:
ERROR

90440 (3)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSCROB

Severity:
ERROR

90440 (4)

Explanation:
The specified file mode number is invalid. It must be a single-digit numeral between 0 and 6.

Module:
DMSEXIFI

Severity:
ERROR

90440 (5)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSEXIST

Severity:
ERROR

90440 (6)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSFILEC

Severity:
ERROR

90440 (7)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSOPBLK

Severity:
ERROR

90440 (8)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSOPDBK

Severity:
ERROR

90440 (9)

Explanation:
The specified file mode number is incorrect. It must be a single-digit numeral between 0 and 6.

Module:
DMSOPEN

Severity:
ERROR

90440 (10)

Explanation:
Incorrect file mode number specified.

Module:
DMSPUSHA

Severity:
ERROR

90440 (11)

Explanation:
The specified file mode is invalid. It must be an alphabetic character.

Module:
DMSQFMOD

Severity:
ERROR

90440 (12)

Explanation:
The specified file mode number is incorrect. It must be a single digit between ‘0’ and ‘6’.

Module:
DMSRENAM

Severity:
ERROR

90440 (13)

Explanation:
The specified file mode number is incorrect. It must be a single digit between ‘0’ and ‘6’.

Module:
DMSVALDT

Severity:
ERROR

90445

Explanation:
Incorrect characters (* or %) were found in directory name part of the *dirname* parameter.

Module:
DMSEXIST

Severity:
ERROR

90450 (1)

Explanation:
Wildcard characters (* or %) were found in either the file name or the file type part of the file ID parameter.

Module:
DMSCATTR

Severity:
ERROR

90450 (2)

Explanation:
Global characters (* or %) were found in either the file name or file type.

Module:
DMSCRALI

Severity:
ERROR

90450 (3)

Explanation:

Characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSCRFIL

Severity:
ERROR

90450 (4)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSCRLOC

Severity:
ERROR

90450 (5)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the file ID.

Module:
DMSCROB

Severity:
ERROR

90450 (6)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSDELOC

Severity:
ERROR

90450 (7)

Explanation:
Global file name characters (* or %) were found in the *fileid* parameter.

Module:
DMSERASE

Severity:
ERROR

90450 (8)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSEXIFI

Severity:
ERROR

90450 (9)

Explanation:
Incorrect characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSEXIST

Severity:
ERROR

90450 (10)

Explanation:
Special characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSFILEC

Severity:
ERROR

90450 (11)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSGRANT

Severity:
ERROR

90450 (12)

Explanation:
Incorrect characters (* or %) were found in either the file name or file type part of the file ID parameter.

Module:
DMSOPBLK

Severity:
ERROR

90450 (13)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSOPDBK

Severity:
ERROR

90450 (14)

Explanation:
Incorrect characters (* or %) were found in either the file name or file type part of the *fn_ft* parameter.

Module:
DMSOPEN

Severity:
ERROR

90450 (15)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fn_ft* parameter.

Module:
DMSQOBJ

Severity:
ERROR

90450 (16)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSRELOC

Severity:
ERROR

90450 (17)

Explanation:
Global file name characters (* or %) were found in either the file name or file type.

Module:
DMSRENAM

Severity:
ERROR

90450 (18)

Explanation:
Incorrect characters (* or %) were found in either the file name or file type.

Module:
DMSREVOK

Severity:
ERROR

90450 (19)

Explanation:
Incorrect characters (* or %) were found in either the file name or file type part of the *fn_ft* parameter.

Module:
DMSTRUNC

Severity:
ERROR

90450 (20)

Explanation:
Wildcard characters (* or %) were found in either the file name or file type part of the *fileid* parameter.

Module:
DMSUDATA

Severity:
ERROR

90450 (21)

Explanation:
There are invalid characters (* or %) in *fn_ft*.

Module:
DMSVALDT

Severity:
ERROR

90455

Explanation:
Incorrect character (* or %) found in file mode or directory fields.

Module:
DMSEXIST

Severity:
ERROR

90460 (1)

Explanation:
File pool IDs in source and target directory names are not the same.

Module:
DMSCRALI

Severity:
ERROR

90460 (2)

Explanation:
File pool IDs in source and target directory are not the same.

Module:
DMSFILEC

Severity:
ERROR

90460 (3)

Explanation:
File pool IDs in source and target directory names are not the same.

Module:
DMSRELOC

Severity:
ERROR

90460 (4)

Explanation:

File pool IDs in source and target directory names are not the same.

Module:
DMSRENAM

Severity:
ERROR

90470

Explanation:
Parameter is not valid; must be FILESPACE, GROUP, or DIRECTORY.

Module:
DMSOPCAT

Severity:
ERROR

90472 (1)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSCATTR

Severity:
ERROR

90472 (2)

Explanation:
Incorrect requestid specified, must be 0 or 1.

Module:
DMSCLBLK

Severity:
ERROR

90472 (3)

Explanation:
Invalid requestid specified, must be 0 or 1.

Module:
DMSCLCAT

Severity:
ERROR

90472 (4)

Explanation:
Invalid *requestid*. It must be 0 or 1.

Module:
DMSCLDBK

Severity:
ERROR

90472 (5)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSCLDIR

Severity:
ERROR

90472 (6)

Explanation:
Invalid *requestid*. It must be 0 or 1.

Module:
DMSCLOSE

Severity:
ERROR

90472 (7)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSCRALI

Severity:
ERROR

90472 (8)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSCRDIR

Severity:
ERROR

90472 (9)

Explanation:
Incorrect *requestid*.

Module:
DMSCRFIL

Severity:
ERROR

90472 (10)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSCRLOC

Severity:
ERROR

90472 (11)

Explanation:
Incorrect *requestid*, must be 0 or 1.

Module:
DMSCROB

Severity:

ERROR

90472 (12)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSDELOC

Severity:
ERROR

90472 (13)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSDIRAT

Severity:
ERROR

90472 (14)

Explanation:
Incorrect request ID specified. It must be 0 or 1.

Module:
DMSDISFS

Severity:
ERROR

90472 (15)

Explanation:
Invalid request ID specified, must be 0 or 1.

Module:
DMSDISSG

Severity:
ERROR

90472 (16)

Explanation:
Incorrect request ID specified, must be 0 or 1.

Module:
DMSENAFS

Severity:
ERROR

90472 (17)

Explanation:
Incorrect requestid specified, must be 0 or 1.

Module:
DMSENASG

Severity:
ERROR

90472 (18)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSERASE

Severity:
ERROR

90472 (19)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSEXIDI

Severity:
ERROR

90472 (20)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSEXIFI

Severity:
ERROR

90472 (21)

Explanation:
Incorrect request ID specified. It must be 0 or 1.

Module:
DMSEXIST

Severity:
ERROR

90472 (22)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSGETDA

Severity:
ERROR

90472 (23)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDD

Severity:
ERROR

90472 (24)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDF

Severity:
ERROR

90472 (25)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDI

Severity:
ERROR

90472 (26)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDK

Severity:
ERROR

90472 (27)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDL

Severity:
ERROR

90472 (28)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDS

Severity:
ERROR

90472 (29)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDT

Severity:
ERROR

90472 (30)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSGETDX

Severity:
ERROR

90472 (31)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSGRANT

Severity:
ERROR

90472 (32)

Explanation:
Incorrect request ID specified, must be 0 or 1.

Module:
DMSOPBLK

Severity:
ERROR

90472 (33)

Explanation:
Requestid is not valid; must be 0 or 1.

Module:
DMSOPCAT

Severity:
ERROR

90472 (34)

Explanation:
requestid must be 0 or 1.

Module:
DMSOPDBK

Severity:
ERROR

90472 (35)

Explanation:
Invalid request ID specified; must be 0 or 1.

Module:
DMSOPDIR

Severity:
ERROR

90472 (36)

Explanation:
requestid must be 0 or 1.

Module:
DMSOPEN

Severity:
ERROR

90472 (37)

Explanation:
Invalid *requestid* specified; must be 0 or 1.

Module:

DMSQCONN

Severity:
ERROR

90472 (38)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSQFPDS

Severity:
ERROR

90472 (39)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSQLIMA

Severity:
ERROR

90472 (40)

Explanation:
Specified *requestid* is invalid; must be 0 or 1.

Module:
DMSQLIMU

Severity:
ERROR

90472 (41)

Explanation:
Invalid *requestid*, must be 0 or 1.

Module:
DMSQOBJ

Severity:
ERROR

90472 (42)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSQUSG

Severity:
ERROR

90472 (43)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSRDBLK

Severity:
ERROR

90472 (44)

Explanation:
Invalid request ID specified, must be 0 or 1.

Module:
DMSRDCAT

Severity:
ERROR

90472 (45)

Explanation:
requestid must be 0 or 1.

Module:
DMSRDDBK

Severity:
ERROR

90472 (46)

Explanation:
requestid must be 0 or 1.

Module:
DMSREAD

Severity:
ERROR

90472 (47)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSRELBK

Severity:
ERROR

90472 (48)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSRELOC

Severity:
ERROR

90472 (49)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSRENAM

Severity:
ERROR

90472 (50)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSREVOK

Severity:
ERROR

90472 (51)

Explanation:
Incorrect request ID specified; it must be 0 or 1.

Module:
DMSTRUNC

Severity:
ERROR

90472 (52)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSUDATA

Severity:
ERROR

90472 (53)

Explanation:
Incorrect request ID specified; must be 0 or 1.

Module:
DMSWRBLK

Severity:
ERROR

90472 (54)

Explanation:
Invalid requestid specified, must be 0 or 1.

Module:
DMSWRCAT

Severity:
ERROR

90472 (55)

Explanation:
Invalid request ID specified, must be 0 or 1.

Module:
DMSWRDBK

Severity:
ERROR

90472 (56)

Explanation:
Incorrect request ID: must be 0 or 1.

Module:
DMSWRITE

Severity:
ERROR

Reason Codes 90476-99631 generated by the VMLIB CSL routines

Explanation, originating module, and severity is documented for reason codes 90476-99631, which are generated by VMLIB CSL routines.

90476 (1)

Explanation:
Invalid file pool ID specified.

Module:
DMSDEUSR

Severity:
ERROR

90476 (2)

Explanation:
Incorrect file pool ID specified.

Module:
DMSDISFS

Severity:
ERROR

90476 (3)

Explanation:
Invalid file pool ID specified.

Module:
DMSDISSG

Severity:
ERROR

90476 (4)

Explanation:
Incorrect file pool ID specified.

Module:
DMSENAFS

Severity:
ERROR

90476 (5)

Explanation:
Incorrect file pool ID specified.

Module:
DMSENASG

Severity:
ERROR

90476 (6)

Explanation:
Invalid file pool ID specified.

Module:
DMSENUSR

Severity:
ERROR

90476 (7)

Explanation:
Incorrect file pool ID specified in the *uniqueid_fpid* parameter.

Module:
DMSEXIST

Severity:
ERROR

90476 (8)

Explanation:
File pool ID is not valid.

Module:
DMSOPCAT

Severity:
ERROR

90476 (9)

Explanation:
Invalid file pool ID specified.

Module:
DMSQCONN

Severity:
ERROR

90476 (10)

Explanation:
Incorrect file pool ID specified.

Module:
DMSQFPDS

Severity:
ERROR

90476 (11)

Explanation:
Invalid file pool ID specified.

Module:
DMSQLIMA

Severity:
ERROR

90476 (12)

Explanation:
Invalid file pool ID specified.

Module:
DMSQLIMU

Severity:
ERROR

90476 (13)

Explanation:
Incorrect file pool ID specified.

Module:
DMSQSFSL

Severity:
ERROR

90476 (14)

Explanation:
Incorrect *filepoolid*.

Module:
DMSQUSG

Severity:
ERROR

90476 (15)

Explanation:
Invalid file pool ID specified.

Module:
DMSRELBK

Severity:
ERROR

90477 (1)

Explanation:
Invalid wait option specified; must be WAIT or NOWAIT.

Module:
DMSCHECK

Severity:
ERROR

90477 (2)

Explanation:
Invalid wait option specified; must be WAIT or NOWAIT.

Module:
DMSSSPTO

Severity:
ERROR

90478

Explanation:
Invalid parameter, must be DETACH or NODETACH.

Module:
DMSDISSG

Severity:
ERROR

90479

Explanation:
Invalid keyword specified, must be DMSOPCAT or DMSCLCAT.

Module:
DMSCPYBF

Severity:
ERROR

90480 (1)

Explanation:
Incorrect lock type, must be SHARE or EXCLUSIVE.

Module:
DMSDISFS

Severity:
ERROR

90480 (2)

Explanation:
Invalid lock type, must be SHARE or EXCLUSIVE.

Module:
DMSDISSG

Severity:
ERROR

90481

Explanation:
Incorrect length specified for the *uniqueid_fpid* parameter. The length must be greater than or equal to 17 and less than or equal to 24.

Module:
DMSEXIST

Severity:
ERROR

90482

Explanation:
Attributes parameter was not specified for a file opened with intent of NEW, WRITE, or REPLACE.

Module:
DMSCLBLK

Severity:
ERROR

90483

Explanation:
WRITE option is not valid when opening a catalog for a directory.

Module:
DMSOPCAT

Severity:
ERROR

90484 (1)

Explanation:
Open type is not valid; must be READ, WRITE, FILEATTR, or READEXT.

Module:
DMSOPCAT

Severity:
ERROR

90484 (2)

Explanation:
WRITE is not a valid Open Data Block type. Must be NEW, READ, or REPLACE.

Module:
DMSOPDBK

Severity:
ERROR

90485 (1)

Explanation:
Invalid buffer length specified.

Module:
DMSCLCAT

Severity:
ERROR

90485 (2)

Explanation:
Invalid buffer length specified.

Module:
DMSCPYBF

Severity:
ERROR

90485 (3)

Explanation:
Incorrect buffer length specified.

Module:
DMSGETER

Severity:
ERROR

90485 (4)

Explanation:
Incorrect buffer length specified.

Module:
DMSGETRS

Severity:
ERROR

90485 (5)

Explanation:
Invalid buffer length specified.

Module:
DMSGETSP

Severity:
ERROR

90485 (6)

Explanation:
Buffer length is not valid.

Module:
DMSOPCAT

Severity:
ERROR

90485 (7)

Explanation:
Invalid length specified for server status parameter.

Module:
DMSQCONN

Severity:
ERROR

90486

Explanation:
An incorrect unique ID—all zeros—specified in the *uniqueid_fpid* parameter.

Module:
DMSEXIST

Severity:
ERROR

90488 (1)

Explanation:
Incorrect number of buffers specified.

Module:
DMSRDBLK

Severity:
ERROR

90488 (2)

Explanation:
Incorrect number of buffers specified.

Module:
DMSWRBLK

Severity:
ERROR

90490 (1)

Explanation:
Invalid number of file space blocks specified.

Module:
DMSENUSR

Severity:
ERROR

90490 (2)

Explanation:
Invalid number of blocks specified.

Module:
DMSRDBLK

Severity:
ERROR

90490 (3)

Explanation:
Number of blocks to read is not greater than zero.

Module:
DMSRDDBK

Severity:
ERROR

90490 (4)

Explanation:
Incorrect number of blocks specified.

Module:
DMSWRBLK

Severity:
ERROR

90490 (5)

Explanation:
Invalid number of blocks specified.

Module:
DMSWRDBK

Severity:
ERROR

90492

Explanation:
Valid parameter must be COMMIT or NOCOMMIT

Module:
DMSCLBLK

Severity:
ERROR

90494 (1)

Explanation

Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSCLBLK

Severity:
ERROR

90494 (2)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSCLDBK

Severity:
ERROR

90494 (3)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSCLOSE

Severity:
ERROR

90494 (4)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSCRDIR

Severity:
ERROR

90494 (5)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSCRFIL

Severity:
ERROR

90494 (6)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSCROB

Severity:
ERROR

90494 (7)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSENUSR

Severity:
ERROR

90494 (8)

Explanation
Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSOPBLK

Severity:
ERROR

90494 (9)

Explanation

Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSOPDBK

Severity:
ERROR

90494 (10)

Explanation

Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSOPEN

Severity:
ERROR

90494 (11)

Explanation

Incorrect date format. The date must be specified in one of the following formats:

- *yy/mm/dd* if SHORTDATE is specified
- *yyyy/mm/dd* if FULLDATE is specified
- *yyyy-mm-dd* if ISODATE is specified

Module:
DMSTRUNC

Severity:
ERROR

90495 (1)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSCLBLK

Severity:
ERROR

90495 (2)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSCLDBK

Severity:
ERROR

90495 (3)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSCLOSE

Severity:
ERROR

90495 (4)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSCRDIR

Severity:
ERROR

90495 (5)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSCRFIL

Severity:
ERROR

90495 (6)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSCROB

Severity:
ERROR

90495 (7)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSENUSR

Severity:
ERROR

90495 (8)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSOPBLK

Severity:
ERROR

90495 (9)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSOPDBK

Severity:
ERROR

90495 (10)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSOPEN

Severity:
ERROR

90495 (11)

Explanation:
Incorrect date specified for *yyyy*, century *yy* portion is restricted to 19 or 20.

Module:
DMSTRUNC

Severity:
ERROR

90496 (1)

Explanation:
Nonnumeric value in date specification.

Module:
DMSCLBLK

Severity:
ERROR

90496 (2)

Explanation:
Nonnumeric value in date specification.

Module:
DMSCLDBK

Severity:
ERROR

90496 (3)

Explanation:
Nonnumeric value in date specification.

Module:
DMSCLOSE

Severity:
ERROR

90496 (4)

Explanation:
Nonnumeric value in date specification.

Module:
DMSCRDIR

Severity:
ERROR

90496 (5)

Explanation:
Incorrect *create_date* or date specified.

Module:
DMSCRFIL

Severity:
ERROR

90496 (6)

Explanation:
Incorrect date specified.

Module:
DMSCROB

Severity:
ERROR

90496 (7)

Explanation:
Nonnumeric value in date specification.

Module:
DMSENUSR

Severity:
ERROR

90496 (8)

Explanation:
Nonnumeric value in date specification.

Module:
DMSOPBLK

Severity:
ERROR

90496 (9)

Explanation:

The date specified is incorrect; it must be a number 0-9.

Module:
DMSOPDBK

Severity:
ERROR

90496 (10)

Explanation:
The date specified is incorrect; it must be a number 0-9.

Module:
DMSOPEN

Severity:
ERROR

90496 (11)

Explanation:
The date specified is incorrect; it must be a number 0-9.

Module:
DMSTRUNC

Severity:
ERROR

90498 (1)

Explanation:
Incorrect time format; must be in the form *hh:mm:ss*.

Module:
DMSCLBLK

Severity:
ERROR

90498 (2)

Explanation:
Incorrect time format; must be in the form *HH:MM:SS*.

Module:
DMSCLDBK

Severity:
ERROR

90498 (3)

Explanation:
Invalid time format; must be in the form *hh:mm:ss*.

Module:
DMSCLOSE

Severity:
ERROR

90498 (4)

Explanation:
Invalid time format; must be in the form *hh:mm:ss*.

Module:
DMSCRDIR

Severity:
ERROR

90498 (5)

Explanation:
Incorrect *create_time* or time format.

Module:
DMSCRFIL

Severity:
ERROR

90498 (6)

Explanation:
Incorrect time format specified.

Module:
DMSCROB

Severity:
ERROR

90498 (7)

Explanation:
Invalid time format; must be in the form HH:MM:SS.

Module:
DMSENUSR

Severity:
ERROR

90498 (8)

Explanation:
Incorrect time format; must be in the form *hh:mm:ss*.

Module:
DMSOPBLK

Severity:
ERROR

90498 (9)

Explanation:
The time specified is in incorrect format; it must be in the format *hh:mm:ss*.

Module:
DMSOPDBK

Severity:
ERROR

90498 (10)

Explanation:
The time specified is in incorrect format; it must be in the format *hh:mm:ss*.

Module:
DMSOPEN

Severity:
ERROR

90498 (11)

Explanation:
The time specified is in incorrect format; it must be in the format *hh:mm:ss*.

Module:
DMSTRUNC

Severity:
ERROR

90499 (1)

Explanation:
Nonnumeric value in time specification.

Module:
DMSCLBLK

Severity:
ERROR

90499 (2)

Explanation:
Nonnumeric value in time specification.

Module:
DMSCLDBK

Severity:
ERROR

90499 (3)

Explanation:
Nonnumeric value in time specification.

Module:
DMSCLOSE

Severity:
ERROR

90499 (4)

Explanation:
Nonnumeric value in time specification.

Module:
DMSCRDIR

Severity:
ERROR

90499 (5)

Explanation:
Incorrect *create_time* or time specified.

Module:
DMSCRFIL

Severity:
ERROR

90499 (6)

Explanation:
Incorrect time specified.

Module:
DMSCROB

Severity:
ERROR

90499 (7)

Explanation:
Nonnumeric value in time specification.

Module:
DMSENUSR

Severity:
ERROR

90499 (8)

Explanation:
Nonnumeric value in time specification.

Module:
DMSOPBLK

Severity:
ERROR

90499 (9)

Explanation:
The time specified is incorrect; it must be a number 0-9.

Module:
DMSOPDBK

Severity:
ERROR

90499 (10)

Explanation:
The time specified is incorrect; it must be a number 0-9.

Module:
DMSOPEN

Severity:
ERROR

90499 (11)

Explanation:
The time specified is incorrect; it must be a number 0-9.

Module:
DMSTRUNC

Severity:
ERROR

90500 (1)

Explanation:
The specified dirname is incorrect.

Module:
DMSCATTR

Severity:
ERROR

90500 (2)

Explanation:
The specified dirname is invalid.

Module:
DMSCRALI

Severity:
ERROR

90500 (3)

Explanation:
The specified dirname is invalid.

Module:
DMSCRDIR

Severity:
ERROR

90500 (4)

Explanation:
The specified *dirname* is incorrect.

Module:
DMSCRFIL

Severity:
ERROR

90500 (5)

Explanation:
The specified director name is incorrect.

Module:
DMSCRLOC

Severity:
ERROR

90500 (6)

Explanation:
The specified *dirname* is incorrect.

Module:
DMSCROB

Severity:
ERROR

90500 (7)

Explanation:
The specified directory name is invalid.

Module:
DMSDELOC

Severity:
ERROR

90500 (8)

Explanation:
The directory name was invalid.

Module:
DMSDIRAT

Severity:
ERROR

90500 (9)

Explanation:
The specified directory name is invalid.

Module:
DMSERASE

Severity:
ERROR

90500 (10)

Explanation:
The specified dirname is invalid.

Module:
DMSEXIDI

Severity:
ERROR

90500 (11)

Explanation:
The specified dirname is invalid.

Module:
DMSEXIFI

Severity:
ERROR

90500 (12)

Explanation:
The specified directory name is incorrect.

Module:
DMSEXIST

Severity:
ERROR

90500 (13)

Explanation:
The specified dirname is incorrect.

Module:
DMSFILEC

Severity:

ERROR

90500 (14)

Explanation:
The specified dirname is invalid.

Module:
DMSGRANT

Severity:
ERROR

90500 (15)

Explanation:
The specified directory name is incorrect.

Module:
DMSOPBLK

Severity:
ERROR

90500 (16)

Explanation:
Directory name is not valid. For BFS file spaces, only the BFS top directory is valid.

Module:
DMSOPCAT

Severity:
ERROR

90500 (17)

Explanation:
The specified directory name is incorrect.

Module:
DMSOPDBK

Severity:
ERROR

90500 (18)

Explanation:
The specified dirname is invalid.

Module:
DMSOPDIR

Severity:
ERROR

90500 (19)

Explanation:
The specified directory name is incorrect.

Module:
DMSOPEN

Severity:
ERROR

90500 (20)

Explanation:
The specified *dirname* is invalid.

Module:
DMSQOBJ

Severity:
ERROR

90500 (21)

Explanation:
The specified dirname is incorrect.

Module:
DMSRELOC

Severity:
ERROR

90500 (22)

Explanation:
The specified directory name is invalid.

Module:
DMSRENAM

Severity:
ERROR

90500 (23)

Explanation:
The specified directory name is incorrect.

Module:
DMSREVOK

Severity:
ERROR

90500 (24)

Explanation:
The specified directory name is incorrect.

Module:
DMSTRUNC

Severity:
ERROR

90500 (25)

Explanation:
The specified dirname is incorrect.

Module:
DMSUDATA

Severity:
ERROR

90500 (26)

Explanation:
The specified directory name is incorrect. If the +/-*filemode.dirid* form of directory ID was specified, the file mode is not accessed or refers to a minidisk.

Module:
DMSVALDT

Severity:
ERROR

90505 (1)

Explanation:
The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSCATTR

Severity:
ERROR

90505 (2)

Explanation:
The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSCRALI

Severity:
ERROR

90505 (3)

Explanation:
The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSCRDIR

Severity:
ERROR

90505 (4)

Explanation:
The specified directory name is of a form that represents an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSCRFIL

Severity:
ERROR

90505 (6)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSCRLOC

Severity:
ERROR

90505 (7)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSCROB

Severity:
ERROR

90505 (8)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSDELOC

Severity:
ERROR

90505 (9)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSDIRAT

Severity:
ERROR

90505 (10)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSERASE

Severity:
ERROR

90505 (11)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSEXIDI

Severity:
ERROR

90505 (12)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSEXIFI

Severity:
ERROR

90505 (13)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSEXIST

Severity:
ERROR

90505 (14)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSFILEC

Severity:
ERROR

90505 (15)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSGRANT

Severity:
ERROR

90505 (16)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSOPBLK

Severity:
ERROR

90505 (17)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSOPCAT

Severity:
ERROR

90505 (18)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSOPDBK

Severity:
ERROR

90505 (19)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSOPDIR

Severity:
ERROR

90505 (20)

Explanation:
The specified directory name is of a form that represents an extended form of directory ID. The directory ID had a form such as ‘+A’. Only directory names or file mode letters are allowed on program function calls.

Module:
DMSOPEN

Severity:
ERROR

90505 (21)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSQOBJ

Severity:

ERROR

90505 (22)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSRELOC

Severity:
ERROR

90505 (23)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSRENAM

Severity:
ERROR

90505 (24)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSREVOK

Severity:
ERROR

90505 (25)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSTRUNC

Severity:
ERROR

90505 (26)

Explanation:
The specified directory name is an extended form of directory ID, such as "+A". Only directory names or file mode letters are allowed on program function calls.

Module:
DMSUDATA

Severity:
ERROR

90510 (1)

Explanation:
The namedef part of the file ID or dirname parameter is no longer than 16 characters.

Module:
DMSCATTR

Severity:
ERROR

90510 (2)

Explanation:
The namedef part of the *fileid* parameter is longer than 16 characters.

Module:
DMSCRALI

Severity:
ERROR

90510 (3)

Explanation:
The namedef parameter is longer than 16 characters.

Module:
DMSCRDIR

Severity:
ERROR

90510 (4)

Explanation:
Syntactic error in *namedef*.

Module:
DMSCRFIL

Severity:
ERROR

90510 (5)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSCRLOC

Severity:
ERROR

90510 (6)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSDELOC

Severity:
ERROR

90510 (7)

Explanation:
The *namedef* parameter is longer than 16 characters.

Module:
DMSDIRAT

Severity:
ERROR

90510 (8)

Explanation:
The *namedef* part of the *dirname* or *fileid* parameter is longer than 16 characters.

Module:
DMSERASE

Severity:
ERROR

90510 (9)

Explanation:
The namedef part of the *dirname* parameter is longer than 16 characters.

Module:
DMSEXIDI

Severity:
ERROR

90510 (10)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSEXIFI

Severity:
ERROR

90510 (11)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSEXIST

Severity:
ERROR

90510 (12)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSFILEC

Severity:
ERROR

90510 (13)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSGRANT

Severity:
ERROR

90510 (14)

Explanation:
The namedef part of the file ID or directory name parameter is longer than 16 characters.

Module:
DMSOPBLK

Severity:
ERROR

90510 (15)

Explanation:
The namedef part of *dirname* is longer than 16 characters.

Module:
DMSOPCAT

Severity:
ERROR

90510 (16)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSOPDBK

Severity:
ERROR

90510 (17)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSOPDIR

Severity:
ERROR

90510 (18)

Explanation:
The namedef part of the *fn_ft* or *dirname* parameter is longer than 16 characters.

Module:
DMSOPEN

Severity:
ERROR

90510 (19)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSRELOC

Severity:
ERROR

90510 (20)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSREVOK

Severity:
ERROR

90510 (21)

Explanation:
The namedef part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSTRUNC

Severity:
ERROR

90510 (22)

Explanation:
The *namedef* part of the *fileid* or *dirname* parameter is longer than 16 characters.

Module:
DMSUDATA

Severity:
ERROR

90510 (23)

Explanation:
A namedef is longer than 16 characters.

Module:
DMSVALDT

Severity:
ERROR

90530 (1)

Explanation:
The namedef part of the file ID or dirname parameter does not exist or was used incorrectly. For example, a namedef that was created for a dirname was used where a file name/file type namedef was expected.

Module:
DMSCATTR

Severity:
ERROR

90530 (2)

Explanation:
The namedef part of the *fileid* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a file ID namedef was expected.

Module:
DMSCRALI

Severity:
ERROR

90530 (3)

Explanation:
The specified namedef does not exist or was used incorrectly. For example, a namedef that was created for a file name/file type was used where a dirname namedef was expected.

Module:
DMSCRDIR

Severity:
ERROR

90530 (4)

Explanation:
Incorrect *namedef* specified.

Module:
DMSCRFIL

Severity:
ERROR

90530 (5)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef containing a file name and file type was expected.

Module:
DMSCRLOC

Severity:
ERROR

90530 (6)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was incorrectly used. For example, a namedef that was created for a directory name was

used where a namedef containing a file name and file type expected.

Module:
DMSDELOC

Severity:
ERROR

90530 (7)

Explanation:
The namedef does not exist or was used incorrectly. For example a namedef for a directory name was used where a namedef for a file name and file type are expected.

Module:
DMSDIRAT

Severity:
ERROR

90530 (8)

Explanation:
The *namedef* part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a temporary name *(namedef)* that was created for a directory name was used where a temporary name for a file was expected.

Module:
DMSERASE

Severity:
ERROR

90530 (9)

Explanation:
The namedef part of the *dirname* parameter does not exist or was used incorrectly. For example, a temporary name (*namedef*) for directory name was used where a temporary name for a file was expected.

Module:
DMSEXIDI

Severity:
ERROR

90530 (10)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a temporary name (*namedef*) for a directory name was used where a temporary name for a file was expected.

Module:
DMSEXIFI

Severity:
ERROR

90530 (11)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSEXIST

Severity:
ERROR

90530 (12)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSFILEC

Severity:
ERROR

90530 (13)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSGRANT

Severity:
ERROR

90530 (14)

Explanation:
The namedef part of the file ID or directory name parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name or file type was expected.

Module:
DMSOPBLK

Severity:
ERROR

90530 (15)

Explanation:
The namedef part of the directory name parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was found where a namedef for a file name and file type was expected.

Module:
DMSOPCAT

Severity:
ERROR

90530 (16)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSOPDBK

Severity:
ERROR

90530 (17)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSOPDIR

Severity:
ERROR

90530 (18)

Explanation

The file ID of the file to be opened is incomplete or incorrect. Some possible errors are:

- The namedef part of the file ID compound parameter (*fn_ft* and *dirname* or *filemode*) does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a *fn_ft* namedef was expected.
- The file ID compound parameter does not resolve to a complete file ID; for example, the file mode was omitted.

Module:
DMSOPEN

Severity:
ERROR

90530 (19)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSRELOC

Severity:
ERROR

90530 (20)

Explanation:
The *namedef* part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a temporary name *(namedef)* that was created for a directory name was used where a temporary name for a file was expected.

Module:
DMSRENAM

Severity:
ERROR

90530 (21)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSREVOK

Severity:
ERROR

90530 (22)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSTRUNC

Severity:
ERROR

90530 (23)

Explanation:
The namedef part of the *fileid* or *dirname* parameter does not exist or was used incorrectly. For example, a namedef that was created for a directory name was used where a namedef for a file name and file type was expected.

Module:
DMSUDATA

Severity:

ERROR

90530 (24)

Explanation

There is an error in the file ID compound parameter. Some possible errors are:

- A namedef does not exist or was used incorrectly.
- The parameter does not resolve to a complete file ID; for example, the file mode was omitted.

Module:
DMSVALDT

Severity:
ERROR

90540 (25)

Explanation:
Incorrect work unit ID specified.

Module:
DMSCATTR

Severity:
ERROR

90540 (26)

Explanation:
Invalid work unit ID. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSCOMM.

Module:
DMSCOMM

Severity:
ERROR

90540 (27)

Explanation:
Specified work unit ID is invalid.

Module:
DMSCRALI

Severity:
ERROR

90540 (28)

Explanation:
Specified work unit ID is invalid.

Module:
DMSCRDIR

Severity:
ERROR

90540 (29)

Explanation:
Specified *workunitid* is incorrect.

Module:
DMSCRFIL

Severity:
ERROR

90540 (30)

Explanation:
Specified work unit ID is invalid.

Module:
DMSCRLOC

Severity:
ERROR

90540 (31)

Explanation:
Specified *workunitid* is incorrect.

Module:
DMSCROB

Severity:
ERROR

90540 (32)

Explanation:
Specified work unit ID is invalid.

Module:
DMSDELOC

Severity:
ERROR

90540 (33)

Explanation:
Specified work unit ID is invalid.

Module:
DMSDEUSR

Severity:
ERROR

90540 (34)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSDISFS

Severity:
ERROR

90540 (35)

Explanation:
Specified work unit ID is invalid.

Module:
DMSDISSG

Severity:

ERROR

90540 (36)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSENAFS

Severity:
ERROR

90540 (37)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSENASG

Severity:
ERROR

90540 (38)

Explanation:
Specified work unit ID is invalid.

Module:
DMSENUSR

Severity:
ERROR

90540 (39)

Explanation:
Specified work unit ID is invalid.

Module:
DMSERASE

Severity:
ERROR

90540 (40)

Explanation:
Specified work unit ID is invalid.

Module:
DMSEXIDI

Severity:
ERROR

90540 (41)

Explanation:
Specified work unit ID is invalid.

Module:
DMSEXIFI

Severity:
ERROR

90540 (42)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSEXIST

Severity:
ERROR

90540 (43)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSFILEC

Severity:
ERROR

90540 (44)

Explanation:
Invalid *workunitid* parameter.

Module:
DMSGETER

Severity:
ERROR

90540 (45)

Explanation:
Specified work unit ID is invalid.

Module:
DMSGETSP

Severity:
ERROR

90540 (46)

Explanation:
Invalid work unit ID.

Module:
DMSGETWU

Severity:
ERROR

90540 (47)

Explanation:
Specified work unit ID is invalid.

Module:
DMSGRANT

Severity:
ERROR

90540 (48)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSOPBLK

Severity:
ERROR

90540 (49)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSOPCAT

Severity:
ERROR

90540 (50)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSOPDBK

Severity:
ERROR

90540 (51)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSOPDIR

Severity:
ERROR

90540 (52)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSOPEN

Severity:
ERROR

90540 (53)

Explanation:
Specified work unit ID is invalid.

Module:
DMSPUSWU

Severity:
ERROR

90540 (54)

Explanation:
Specified work unit ID is invalid.

Module:
DMSQCONN

Severity:
ERROR

90540 (55)

Explanation:
Incorrect work unit ID was specified.

Module:
DMSQFPDS

Severity:
ERROR

90540 (56)

Explanation:
Specified work unit ID is invalid.

Module:
DMSQLIMA

Severity:
ERROR

90540 (57)

Explanation:
Specified work unit ID is invalid.

Module:
DMSQLIMU

Severity:
ERROR

90540 (58)

Explanation:
Specified *workunitid* is invalid.

Module:
DMSQOBJ

Severity:
ERROR

90540 (59)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSQUSG

Severity:
ERROR

90540 (60)

Explanation:
Incorrect *workunitid* parameter.

Module:
DMSREG

Severity:
ERROR

90540 (61)

Explanation:
Invalid work unit ID specified.

Module:

DMSRELBK

Severity:
ERROR

90540 (62)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSRELOC

Severity:
ERROR

90540 (63)

Explanation:
Specified work unit ID is invalid.

Module:
DMSRENAM

Severity:
ERROR

90540 (64)

Explanation:
Invalid work unit ID. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSRETWU.

Module:
DMSRETWU

Severity:
ERROR

90540 (65)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSREVOK

Severity:
ERROR

90540 (66)

Explanation:
Invalid work unit ID. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSROLLB.

Module:
DMSROLLB

Severity:
ERROR

90540 (67)

Explanation:
Invalid work unit ID.

Module:
DMSSETAG

Severity:
ERROR

90540 (68)

Explanation:
Invalid work unit ID.

Module:
DMSSSPTO

Severity:
ERROR

90540 (69)

Explanation:
Specified work unit ID is incorrect.

Module:
DMSTRUNC

Severity:
ERROR

90540 (70)

Explanation:
Specified work unit ID is invalid.

Module:
DMSWRACC

Severity:
ERROR

90545

Explanation:
System error. The work unit ID could not be associated with a user ID.

Module:
Common

Severity:
ERROR

90550

Explanation:
No work unit IDs are available. You must re-IPL your virtual machine to make some work unit IDs available.

Module:
DMSGETWU

Severity:
ERROR

90555

Explanation:
0 was specified for the request ID, but there are no active asynchronous requests.

Module:
DMSCHECK

Severity:
ERROR

90570

Explanation:
Request ID has already been marked. Any optional data words have not been modified by DMSMARK.

Module:
DMSMARK

Severity:
WARNING

90590 (1)

Explanation:
There is no default file pool currently defined, and the file pool ID was not specified as part of the dirname.

Module:
DMSCATTR

Severity:
ERROR

90590 (2)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSCRALI

Severity:
ERROR

90590 (3)

Explanation:
There is no default file pool currently defined, and file pool ID was not specified as part of the dirname.

Module:
DMSCRDIR

Severity:
ERROR

90590 (4)

Explanation:
There is no default file pool currently defined, and file pool ID was not specified as part of *dirname*.

Module:
DMSCRFIL

Severity:
ERROR

90590 (5)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSCRLOC

Severity:
ERROR

90590 (6)

Explanation:
There is no default file pool currently defined, and file pool ID was not specified as part of the *dirname*.

Module:
DMSCROB

Severity:
ERROR

90590 (7)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSDELOC

Severity:
ERROR

90590 (8)

Explanation:
There is no default file pool defined and the file pool ID was not specified as part of the directory name.

Module:
DMSDIRAT

Severity:
ERROR

90590 (9)

Explanation:
There is no default file pool currently defined and the file pool ID was not specified as part of the directory name.

Module:
DMSERASE

Severity:
ERROR

90590 (10)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSEXIDI

Severity:
ERROR

90590 (11)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSEXIFI

Severity:
ERROR

90590 (12)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSEXIST

Severity:
ERROR

90590 (13)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSFILEC

Severity:
ERROR

90590 (14)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSGRANT

Severity:
ERROR

90590 (15)

Explanation:
There is no default file pool currently defined, and file pool ID was not specified as part of the directory name.

Module:
DMSOPBLK

Severity:
ERROR

90590 (16)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.

Module:
DMSOPDBK

Severity:
ERROR

90590 (17)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSOPDIR

Severity:
ERROR

90590 (18)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.

Module:
DMSOPEN

Severity:
ERROR

90590 (19)

Explanation:
There is no default file pool currently defined, and file pool ID was not specified as part of the directory name.

Module:
DMSQOBJ

Severity:
ERROR

90590 (20)

Explanation:
There is no default file pool currently defined, and file pool ID was not specified as part of the dirname.

Module:
DMSRELOC

Severity:
ERROR

90590 (21)

Explanation:
There is no default file pool defined and the file pool ID was not specified as part of the directory name.

Module:
DMSRENAM

Severity:
ERROR

90590 (22)

Explanation:

There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.

Module:
DMSREVOK

Severity:
ERROR

90590 (23)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the directory name.

Module:
DMSTRUNC

Severity:
ERROR

90590 (24)

Explanation:
There is no default file pool currently defined, and *filepoolid* was not specified as part of the dirname.

Module:
DMSUDATA

Severity:
ERROR

90590 (25)

Explanation:
There is no default file pool currently defined, and the file pool ID was not specified as part of the directory name.

Module:
DMSVALDT

Severity:
ERROR

90600

Explanation:
No free file modes available.

Module:
DMSGETFM

Severity:
ERROR

90601 (1)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSCATTR

Severity:
ERROR

90601 (2)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSCRALI

Severity:
ERROR

90601 (3)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSCRDIR

Severity:
ERROR

90601 (4)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSCRFIL

Severity:
ERROR

90601 (5)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSCRLOC

Severity:
ERROR

90601 (6)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSCROB

Severity:
ERROR

90601 (7)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:

DMSDELOC

Severity:
ERROR

90601 (8)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSDIRAT

Severity:
ERROR

90601 (9)

Explanation:
Provided only a file mode letter as input but it corresponds to a minidisk. An option or parameter specified is invalid with a minidisk.

Module:
DMSERASE

Severity:
ERROR

90601 (10)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSEXIDI

Severity:
ERROR

90601 (11)

Explanation:
Exist for directory function cannot be performed on a minidisk.

Module:
DMSEXIST

Severity:
ERROR

90601 (12)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSGRANT

Severity:
ERROR

90601 (13)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSOPBLK

Severity:
ERROR

90601 (14)

Explanation:
Directory is on a CMS minidisk and open intent was not FILE.

Module:
DMSOPDIR

Severity:
ERROR

90601 (15)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSQOBJ

Severity:
ERROR

90601 (16)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSRELOC

Severity:
ERROR

90601 (17)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSREVOK

Severity:
ERROR

90601 (18)

Explanation:
Input file mode letter did not represent an accessed SFS directory.

Module:
DMSUDATA

Severity:
ERROR

90602 (1)

Explanation:
Incorrect file mode specified. Asterisk only allowed for open intent of READ.

Module:
DMSOPDBK

Severity:
ERROR

90602 (2)

Explanation:
Incorrect file mode specified. Asterisk is only allowed for open intent of READ.

Module:
DMSOPEN

Severity:
ERROR

90603 (1)

Explanation:
Attempted to open the file for output, but the file mode is read-only.

Module:
DMSFILEC

Severity:
ERROR

90603 (2)

Explanation:
Attempted to open the file for output and file mode is read-only.

Module:
DMSOPDBK

Severity:
ERROR

90603 (3)

Explanation:
Attempted to open the file for output and file mode is read/only.

Module:
DMSOPEN

Severity:
ERROR

90603 (4)

Explanation:
Attempted to truncate the file and the file mode is read-only.

Module:
DMSTRUNC

Severity:
ERROR

90604 (1)

Explanation:
Minidisk file has already been opened with the FS macro interface.

Module:
DMSFILEC

Severity:
ERROR

90604 (2)

Explanation:
You have already opened the SFS file with the FS macro interface for output.

Module:
DMSOPBLK

Severity:
ERROR

90604 (3)

Explanation:
Minidisk file already open using the FS macro interface.

Module:
DMSOPDBK

Severity:
ERROR

90604 (4)

Explanation:
You have already opened the file with the FS macro interface: If it is a minidisk file, it is open for input or output; if it is an SFS file, it is opened for output.

Module:
DMSOPEN

Severity:
ERROR

90604 (5)

Explanation:
Minidisk file already open using the FS macro interface.

Module:
DMSTRUNC

Severity:
ERROR

90605

Explanation:
An attempt was made to associate an ACF with a minidisk file.

Module:
DMSOPDBK

Severity:
ERROR

90606 (1)

Explanation:
I/O error found when processing a minidisk file.

Module:
DMSOPDBK

Severity:
ERROR

90606 (2)

Explanation:
I/O error found when processing a minidisk file.

Module:
DMSOPEN

Severity:
ERROR

90606 (3)

Explanation:
I/O error found when processing a minidisk file.

Module:
DMSTRUNC

Severity:
ERROR

90610

Explanation:
Source file was empty and the target file is a minidisk.

Module:
DMSFILEC

Severity:
ERROR

90611 (1)

Explanation:
Input file mode letter did not represent an accessed SFS directory in the second file ID.

Module:
DMSCRALI

Severity:
ERROR

90611 (2)

Explanation:
Input file mode letter did not represent an accessed SFS directory in the second file ID.

Module:
DMSRELOC

Severity:
ERROR

90614 (1)

Explanation:
Erase attempted on a read/only minidisk file.

Module:
DMSERASE

Severity:
ERROR

90614 (2)

Explanation:
Rename attempted on a read/only minidisk file.

Module:
DMSRENAM

Severity:
ERROR

90615

Explanation:
The specified disk must be a CMS-formatted minidisk.

Module:
DMSTRUNC

Severity:
ERROR

90617 (1)

Explanation:
The input file was erased. This occurs for a minidisk file when the number of records is zero. For an SFS file, this occurs when the number of records is zero and the ALLOWEMPTY parameter was not specified when the file was opened using DMSOPDBK and blocks have been written to the file.

Module:
DMSCLDBK

Severity:
WARNING

90617 (2)

Explanation:
The minidisk file was erased because it was opened with an intent of REPLACE, but no records were written to it.

Module:
DMSCLOSE

Severity:
WARNING

90620 (1)

Explanation:

RECOVER option was specified for a minidisk file. All minidisk files are nonrecoverable.

Module:
DMSOPDBK

Severity:
WARNING

90620 (2)

Explanation:
RECOVER parameter was specified for a minidisk file. All minidisk files are nonrecoverable.

Module:
DMSOPEN

Severity:
WARNING

90621 (1)

Explanation:
INPLACE option was specified for a minidisk file with a file mode number other than 6 or NOTINPLACE was specified for a minidisk file with a file mode number of 6.

Module:
DMSOPDBK

Severity:
WARNING

90621 (2)

Explanation:
INPLACE parameter was specified for a minidisk file with a file mode number other than 6 or NOTINPLACE was specified for a minidisk file with a file mode number of 6.

Module:
DMSOPEN

Severity:
WARNING

90622 (1)

Explanation:
ALLOWEMPTY option was specified for a minidisk file.

Module:
DMSOPDBK

Severity:
WARNING

90622 (2)

Explanation:
ALLOWEMPTY parameter was specified for a minidisk file.

Module:
DMSOPEN

Severity:
WARNING

90622 (3)

Explanation:
ALLOWEMPTY parameter was specified for a minidisk file.

Module:
DMSTRUNC

Severity:
WARNING

90623

Explanation:
OLDDATEREF | NEWDATEREF was specified for a minidisk file.

Module:
DMSFILEC

Severity:
WARNING

90640

Explanation:
Incorrect method. Value must be 0, 1, or 2.

Module:
DMSPOINT

Severity:
ERROR

90641

Explanation:
Incorrect *new_read_offset*. Resulting read pointer must be between 1 and 2^{31}-1.

Module:
DMSPOINT

Severity:
ERROR

90642

Explanation:
Incorrect *new_write_offset*. Resulting write pointer must be between 1 and 2^{31}-1.

Module:
DMSPOINT

Severity:
ERROR

90680 (1)

Explanation:
I/O error accessing OS dataset.

Module:
DMSEXIFI

Severity:
ERROR

90680 (2)

Explanation:
I/O error accessing OS dataset.

Module:
DMSEXIST

Severity:
ERROR

90680 (3)

Explanation:
I/O error accessing OS dataset.

Module:
DMSOPDBK

Severity:
ERROR

90680 (4)

Explanation:
I/O error accessing OS dataset.

Module:
DMSOPEN

Severity:
ERROR

90680 (5)

Explanation:
I/O error accessing OS dataset.

Module:
DMSTRUNC

Severity:
ERROR

90681 (1)

Explanation:
OS read password protected dataset.

Module:
DMSEXIFI

Severity:
ERROR

90681 (2)

Explanation:
OS read password protected dataset.

Module:
DMSEXIST

Severity:
ERROR

90681 (3)

Explanation:
OS read password protected dataset.

Module:
DMSOPDBK

Severity:
ERROR

90681 (4)

Explanation:
OS read password protected dataset.

Module:
DMSOPEN

Severity:
ERROR

90681 (5)

Explanation:
OS read password protected dataset.

Module:
DMSTRUNC

Severity:
ERROR

90682 (1)

Explanation:
OS dataset organization is not BSAM, QSAM or BPAM.

Module:
DMSEXIFI

Severity:
ERROR

90682 (2)

Explanation:
OS dataset organization is not BSAM, QSAM or BPAM.

Module:
DMSEXIST

Severity:
ERROR

90682 (3)

Explanation:
OS dataset organization is not BSAM, QSAM or BPAM.

Module:
DMSOPDBK

Severity:
ERROR

90682 (4)

Explanation:
OS dataset organization is not BSAM, QSAM or BPAM.

Module:

DMSOPEN

Severity:
ERROR

90682 (5)

Explanation:
OS dataset organization is not BSAM, QSAM or BPAM.

Module:
DMSTRUNC

Severity:
ERROR

90683 (1)

Explanation:
OS dataset more than 16 extents.

Module:
DMSEXIFI

Severity:
ERROR

90683 (2)

Explanation:
OS dataset more than 16 extents.

Module:
DMSEXIST

Severity:
ERROR

90683 (3)

Explanation:
OS dataset more than 16 extents.

Module:
DMSOPDBK

Severity:
ERROR

90683 (4)

Explanation:
OS dataset more than 16 extents.

Module:
DMSOPEN

Severity:
ERROR

90683 (5)

Explanation:
OS dataset more than 16 extents.

Module:
DMSTRUNC

Severity:
ERROR

90684 (1)

Explanation:
Attempted to open a file on an OS- or DOS-formatted minidisk.

Module:
DMSFILEC

Severity:
ERROR

90684 (2)

Explanation:
Attempt to open a file on an OS or DOS formatted minidisk.

Module:
DMSOPDBK

Severity:
ERROR

90684 (3)

Explanation:
Attempt to open a file on an OS or DOS formatted minidisk.

Module:
DMSOPEN

Severity:
ERROR

90684 (4)

Explanation:
Attempt to open a file on an OS or DOS formatted minidisk.

Module:
DMSTRUNC

Severity:
ERROR

90685 (1)

Explanation:
System error. Unexpected error returned when erasing a minidisk file.

Module:
DMSERASE

Severity:
ERROR

90685 (2)

Explanation:
Received an unexpected return code during a search for a minidisk file.

Module:
DMSEXIFI

Severity:
ERROR

90685 (3)

Explanation:
Received an unexpected return code during a search for a minidisk file.

Module:
DMSEXIST

Severity:
ERROR

90685 (4)

Explanation:
Received an unexpected return code while opening a minidisk file.

Module:
DMSOPDBK

Severity:
ERROR

90685 (5)

Explanation:
Received an unexpected return code while opening a minidisk file.

Module:
DMSOPEN

Severity:
ERROR

90685 (6)

Explanation:
Received an unexpected return code while opening a minidisk file.

Module:
DMSTRUNC

Severity:
ERROR

90690

Explanation:
lrecl and *numrecs* are required parameters when the file has been opened for NEW or REPLACE.

Module:
DMSCLDBK

Severity:
ERROR

90691

Explanation:
The input value for the number of records is greater than the number of records written to a variable format file, or the input value for the logical record length does not match the length of the longest record actually written to a variable format file.

Module:
DMSCLDBK

Severity:
WARNING

90700 (1)

Explanation:
The input file ID resolves to a file mode letter and it is not accessed.

Module:
Common

Severity:
ERROR

90700 (2)

Explanation:
Input in *dirid*, *filemode*, or *namedef2* was an unaccessed file mode.

Module:
DMSVALDT

Severity:
WARNING

94000

Explanation:
System error in module DMS2RQ.

Module:
Common

Severity:
ERROR

95100

Explanation:
System error. Invalid request type passed to SFS.

Module:
Common

Severity:
ERROR

95200 (1)

Explanation:
A fatal communication error occurred during a previous request. Any attempt to communicate with a file pool server will be rejected until your virtual machine is re-IPLed.

Module:
Common

Severity:
ERROR

95200 (2)

Explanation:
System error. Further attempts to access the CRR recovery server will be rejected.

Module:
DMSCHREG

Severity:
ERROR

95200 (3)

Explanation:
System error. Further attempts to access a CRR recovery server will be rejected.

Module:
DMSGETRS

Severity:
ERROR

95200 (4)

Explanation:
System error. Further attempts to access a CRR recovery server will be rejected.

Module:
DMSREG

Severity:
ERROR

95300

Explanation:
System error. An invalid operation was requested internally by SFS.

Module:
Common

Severity:
ERROR

95400 (1)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSCATTR

Severity:
ERROR

95400 (2)

Explanation:
The specified work unit was already active for the specified file pool when DMSCRLOC was executed.

Module:
DMSCRLOC

Severity:
ERROR

95400 (3)

Explanation:
The specified work unit was already active for the specified file pool when DMSDELOC was executed.

Module:
DMSDELOC

Severity:
ERROR

95400 (4)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSDEUSR

Severity:
ERROR

95400 (5)

Explanation:
The specified work unit was already active for the specified file pool when DMSDIRAT was executed.

Module:
DMSDIRAT

Severity:
ERROR

95400 (6)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSDISFS

Severity:
ERROR

95400 (7)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSDISSG

Severity:
ERROR

95400 (8)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSENAFS

Severity:
ERROR

95400 (9)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSENASG

Severity:
ERROR

95400 (10)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSENUSR

Severity:
ERROR

95400 (11)

Explanation:
A logical unit of work is already in process for the specified work unit ID.

Module:
DMSOPCAT

Severity:
ERROR

95400 (12)

Explanation:
A logical unit of work is being processed for the specified work unit and file pool ID.

Module:
DMSQFPDS

Severity:
ERROR

95400 (13)

Explanation:
This work unit was already active for the specified file pool when DMSQLIMU was executed.

Module:
DMSQLIMU

Severity:
ERROR

95400 (14)

Explanation:
Logical unit of work is already in process for the specified *workunitid*.

Module:
DMSQUSG

Severity:
ERROR

95400 (15)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSRELBK

Severity:
ERROR

95400 (16)

Explanation:
This work unit was already active for the specified file pool when DMSRELOC was executed.

Module:
DMSRELOC

Severity:
ERROR

95400 (17)

Explanation:
A logical unit of work is already in process for this file pool for the specified work unit ID.

Module:
DMSWRACC

Severity:
ERROR

95500

Explanation:
Intent was WRITE, and you have made uncommitted changes to another file pool for the specified work unit ID.

Module:
DMSOPCAT

Severity:
ERROR

95600 (1)

Explanation:
You have another file pool object open and specified COMMIT.

Module:
DMSCLBLK

Severity:
ERROR

95600 (2)

Explanation:
You have another file pool object open for the specified work unit.

Module:
DMSCLCAT

Severity:
ERROR

95600 (3)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSCLDBK

Severity:
ERROR

95600 (4)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSCLDIR

Severity:
ERROR

95600 (5)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSCLOSE

Severity:
ERROR

95600 (6)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement. This is an SFS-specific reason code.

Module:
DMSCOMM

Severity:
ERROR

95600 (7)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSCRALI

Severity:
ERROR

95600 (8)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSCRDIR

Severity:
ERROR

95600 (9)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file or directory is open and the file pool server does not have the Commit Without Close enhancement. This is an SFS-specific reason code.

Module:
DMSCRFIL

Severity:
ERROR

95600 (10)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSCROB

Severity:
ERROR

95600 (11)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSERASE

Severity:
ERROR

95600 (12)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSEXIDI

Severity:
ERROR

95600 (13)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSEXIFI

Severity:
ERROR

95600 (14)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSEXIST

Severity:
ERROR

95600 (15)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSFILEC

Severity:
ERROR

95600 (16)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSGRANT

Severity:
ERROR

95600 (17)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSRENAM

Severity:
ERROR

95600 (18)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement. This is an SFS-specific reason code.

Module:
DMSRETWU

Severity:
ERROR

95600 (19)

Explanation:
The work unit was not committed. This could occur because an open file was modified with Write Blocks or a file in the directory is open and the file pool server does not have the Commit Without Close enhancement.

Module:
DMSREVOK

Severity:
ERROR

95700 (1)

Explanation:
System error. No open file pool object found for the specified token.

Module:
DMSCLBLK

Severity:
ERROR

95700 (2)

Explanation:
System error. No open file pool object found for the specified token.

Module:
DMSCLCAT

Severity:
ERROR

95700 (3)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSCLDBK

Severity:
ERROR

95700 (4)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSCLDIR

Severity:
ERROR

95700 (5)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSCLOSE

Severity:
ERROR

95700 (6)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSCRALI

Severity:
ERROR

95700 (7)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSCRDIR

Severity:
ERROR

95700 (8)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSERASE

Severity:
ERROR

95700 (9)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSEXIDI

Severity:
ERROR

95700 (10)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSEXIFI

Severity:
ERROR

95700 (11)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSEXIST

Severity:
ERROR

95700 (12)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSFILEC

Severity:
ERROR

95700 (13)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDA

Severity:
ERROR

95700 (14)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDD

Severity:
ERROR

95700 (15)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDF

Severity:
ERROR

95700 (16)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDI

Severity:
ERROR

95700 (17)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDK

Severity:
ERROR

95700 (18)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDL

Severity:
ERROR

95700 (19)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDS

Severity:
ERROR

95700 (20)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDT

Severity:
ERROR

95700 (21)

Explanation:
System error. No open directory found for the specified token.

Module:
DMSGETDX

Severity:
ERROR

95700 (22)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSGRANT

Severity:
ERROR

95700 (23)

Explanation:
No open file pool object found for the specified token.

Module:
DMSRDBLK

Severity:
ERROR

95700 (24)

Explanation:
No open file pool object found for the specified token.

Module:
DMSRDCAT

Severity:
ERROR

95700 (25)

Explanation:
System error. No open file was found for internal token passed to SFS.

Module:
DMSRDDBK

Severity:
ERROR

95700 (26)

Explanation:
System error. No open file was found for internal token passed to SFS.

Module:
DMSREAD

Severity:
ERROR

95700 (27)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSRENAM

Severity:
ERROR

95700 (28)

Explanation:
System error. COMMIT was specified. No open file found for internal token passed to SFS.

Module:
DMSREVOK

Severity:
ERROR

95700 (29)

Explanation:
System error. No open file pool object found for the specified token.

Module:
DMSWRBLK

Severity:
ERROR

95700 (30)

Explanation:
No open file pool object found for the specified token.

Module:
DMSWRCAT

Severity:
ERROR

95700 (31)

Explanation:
System error. No open file found for internal token passed to SFS.

Module:
DMSWRDBK

Severity:
ERROR

95700 (32)

Explanation:
System error. No open file found for internal token passed to SFS.

Module:
DMSWRITE

Severity:
ERROR

95750 (1)

Explanation:
No file opened with DMSOPDBK found for the specified token.

Module:
DMSCLDBK

Severity:
ERROR

95750 (2)

Explanation:
No file opened by DMSOPEN found for the specified token.

Module:
DMSCLOSE

Severity:
ERROR

95750 (3)

Explanation:
Incorrect token specified. File not open or not opened using DMSOPEN.

Module:
DMSPOINT

Severity:
ERROR

95750 (4)

Explanation:
No file opened using DMSOPDBK was found for the specified token.

Module:
DMSRDDBK

Severity:
ERROR

95750 (5)

Explanation:
No file opened using DMSOPEN was found for the specified token.

Module:
DMSREAD

Severity:
ERROR

95750 (6)

Explanation:
No file opened using DMSOPDBK found for the specified token.

Module:
DMSWRDBK

Severity:
ERROR

95750 (7)

Explanation:
No file opened using DMSOPEN found for the specified token.

Module:
DMSWRITE

Severity:
ERROR

95777

Explanation:
The ERASE function issued by DMSCLDBK for a file mode number 3 file failed.

Module:
DMSCLDBK

Severity:
ERROR

95800

Explanation:
System error. Incorrect work unit range passed to SFS.

Module:
DMSPURWU

Severity:
ERROR

95900

Explanation:
Logical unit of work in process for an active work unit ID and FORCE option not specified. No communication paths were severed.

Module:
DMSPURWU

Severity:
ERROR

95950

Explanation:
Logical unit of work was in process for an active work unit ID and FORCE option was specified. Any active work has been rolled back.

Module:
DMSPURWU

Severity:
WARNING

96100 (1)

Explanation:
Insufficient virtual storage for a get storage request from a user's virtual machine.

Module:
Common

Severity:
ERROR

96100 (2)

Explanation:
Insufficient virtual storage. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSCOMM. This request was ignored.

Module:
DMSCOMM

Severity:
ERROR

96100 (3)

Explanation:
Insufficient virtual storage. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSRETWU. This request was ignored.

Module:
DMSRETWU

Severity:
ERROR

96100 (4)

Explanation:
Insufficient virtual storage. The work unit was not rolled back or committed. All protected resources are in the same state they were prior to the call to DMSROLLB. This request was ignored.

Module:
DMSROLLB

Severity:
ERROR

96100 (5)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPCC

Severity:
ERROR

96100 (6)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPCD

Severity:
ERROR

96100 (7)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPCI

Severity:
ERROR

96100 (8)

Explanation:
Insufficient CMS virtual storage was available for this request.

Module:
DMSSPCP

Severity:
ERROR

96100 (9)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPCQ

Severity:
ERROR

96100 (10)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPCR

Severity:
ERROR

96100 (11)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPCRP

Severity:
ERROR

96100 (12)

Explanation:
Insufficient CMS virtual storage was available to perform the service.

Module:
DMSSPLA

Severity:
ERROR

96100 (13)

Explanation:
Insufficient CMS storage was available to perform the service.

Module:
DMSSPLR

Severity:
ERROR

96100 (14)

Explanation:
Insufficient virtual storage.

Module:
DMSSSPTO

Severity:
ERROR

96100 (15)

Explanation:
Insufficient virtual storage for a get storage request from user's virtual machine on error.

Module:

DMSTCD

Severity:
ERROR

96100 (16)

Explanation:
CMS was unable to obtain storage.

Module:
StackBufferCreate—DMSSTKC

Severity:
ERROR

96100 (17)

Explanation:
CMS could not obtain storage. Nothing was placed on the program stack.

Module:
StackWrite—DMSSTKW

Severity:
ERROR

96200

Explanation:
System error in storage management while trying to acquire virtual storage in a user's virtual machine.

Module:
Common

Severity:
ERROR

96300

Explanation:
Error while trying to release virtual storage in a user's machine. Further attempts to access the file pool will be rejected.

Module:
DMSOPEN

Severity:
ERROR

96400

Explanation:
System error. Error in APPC/VM IDENTIFY function.

Module:
Common

Severity:
ERROR

96500 (1)

Explanation:
COMMIT was specified, and there is an asynchronous request in process for the specified work unit.

Module:
DMSCLBLK

Severity:
ERROR

96500 (2)

Explanation:
There is an asynchronous request in process for the specified work unit.

Module:
DMSCLCAT

Severity:
ERROR

96600

Explanation:
Error from CSL when attempting to call the SFS user accounting exit.

Module:
Common

Severity:
ERROR

96610 (1)

Explanation:
A CSL routine required by this routine was dropped or not loaded.

Module:
Common

Severity:
ERROR

96610 (2)

Explanation:
CSL error calling DMSSETAG Set Transaction Tag. Routine dropped or not loaded.

Module:
DMSGETWU

Severity:
ERROR

96620

Explanation:
System error. This routine called another CSL routine with an incorrect number of parameters.

Module:
Common

Severity:
ERROR

96700

Explanation:

Request issued while a commit is in process for the specified work unit ID.

Module:
Common

Severity:
ERROR

96800

Explanation:
One or more files or directories are open for the specified work unit ID.

Module:
DMSCLCAT

Severity:
ERROR

96802 (1)

Explanation:
One or more nonrecoverable files could not be closed.

Module:
DMSRETWU

Severity:
WARNING

96802 (2)

Explanation:
One or more nonrecoverable files could not be closed.

Module:
DMSROLLB

Severity:
WARNING

97100

Explanation:
Nonzero return code received from SFS user accounting exit.

Module:
Common

Severity:
ERROR

97200

Explanation:
System error detected by APPC/VM.

Module:
Common

Severity:
ERROR

97250

Explanation:
You have attempted to establish more APPC/VM connections than the maximum allowed for your virtual machine, as determined by the MAXCONN value in your CP directory.

Module:
Common

Severity:
ERROR

97280 (1)

Explanation:
Your attempt exceeds the number of APPC/VM connections allowed for the file pool.

Module:
Common

Severity:
ERROR

97280 (2)

Explanation:
Your attempt exceeds the number of APPC/VM connections allowed for the CRR recovery server.

Module:
DMSCHREG

Severity:
ERROR

97280 (3)

Explanation:
Your attempt exceeds the number of APPC/VM connections allowed for the CRR recovery server.

Module:
DMSGETRS

Severity:
ERROR

97280 (4)

Explanation:
Your attempt exceeds the number of APPC/VM connections allowed for the CRR recovery server.

Module:
DMSREG

Severity:
ERROR

97290

Explanation:
You have attempted to establish more APPC/VM connections than the maximum allowed for your virtual machine, as determined by the MAXCONN value in your CP directory. An inactive path has been severed and another attempt will be made to establish an APPC/VM connection for your request.

Module:
Common

Severity:
WARNING

97300

Explanation:
An attempt was made to use several file pool identifiers that resolve to the same file pool ID.

Module:
Common

Severity:
ERROR

97400

Explanation:
Sever condition returned from APPC/VM communication request. If your application receives this reason code intermittently, but the file pool is still available and other commands work correctly, the file pool server may be improperly configured. Notify your file pool administrator of this condition. (Ask the administrator to check the USERS start-up parameter value.)

Module:
Common

Severity:
ERROR

97450

Explanation:
Sever condition returned from communication request. Accessed directories for lost file pools were released.

Module:
Common

Severity:
ERROR

97480

Explanation:
Communication path severed due to an error detected by a file pool server. This code will appear only as part of the work unit error information that is provided when you specify the *wuerror* parameter.

Module:
Common

Severity:
ERROR

97500

Explanation:
Specified file pool is unavailable (no resource identified to APPC/VM for the specified file pool ID).

Module:
Common

Severity:
ERROR

97600

Explanation:
A function other than DMSCHECK was invoked while an outstanding asynchronous request was in process for the specified file pool ID and work unit ID.

Module:
Common

Severity:
ERROR

97700

Explanation:
A DMSCHECK request was issued when no asynchronous request was in process for the specified file pool ID and work unit ID.

Module:
Common

Severity:
ERROR

97800

Explanation:
For a DMSCHECK request passed to SFS, the request type did not match the request type of the outstanding asynchronous request.

Module:
Common

Severity:
ERROR

98000

Explanation:
System error. Unexpected return code or reason code from synchronization point manager routine DMSREG.

Module:
Common

Severity:
ERROR

98100

Explanation:
System error. Unexpected return code or reason code from synchronization point manager routine DMSCHREG.

Module:
Common

Severity:

ERROR

98200

Explanation:
System error. Unexpected return code or reason code from synchronization point manager routine DMSGETRS.

Module:
Common

Severity:
ERROR

98300

Explanation:
System error. Unexpected return code or reason code from synchronization point manager routine.

Module:
Common

Severity:
ERROR

98400

Explanation:
System error. COMMIT was specified. File attributes were not updated for a file that was written to.

Module:
Common

Severity:
ERROR

98500

Explanation:
System error. An attempt to call a system CSL routine resulted in an unexpected return code from CSL.

Module:
Common

Severity:
ERROR

98550

Explanation:
System error. SFS returned an unexpected warning.

Module:
Common

Severity:
WARNING

98600

Explanation:
An attempt to write to a file pool was rejected because only one write-mode resource is allowed for this unit of work, and another resource is already in write mode.

Module:
Common

Severity:
ERROR

98700 (1)

Explanation:
The target file pool server does not support the requested function. (The file pool server is not at the proper release or service level.)

Module:
Common

Severity:
ERROR

98700 (2)

Explanation:
The UNRESOLVED keyword is not supported by this file pool. However, if the base file and the necessary authorizations exist, the alias was created.

Module:
DMSCRALI

Severity:
WARNING

98700 (3)

Explanation:
Server is not at a service level that supports the KEEPAUTH parameter.

Module:
DMSDEUSR

Severity:
ERROR

98700 (4)

Explanation:
File pool is at a release level that does not support enable for a *function* request.

Module:
DMSENAFS

Severity:
ERROR

98700 (5)

Explanation:
Server is not at a service level that supports BFS file spaces.

Module:
DMSENUSR

Severity:
ERROR

98700 (6)

Explanation:
DATAONLY option is not supported by this file pool. The file pool server is at the z/VM Version 1 Release 1.0 or earlier level.

Module:
DMSERASE

Severity:
ERROR

98700 (7)

Explanation

Server managing this file does not support certain parameters of DMSOPBLK:

NORECOVER
RECOVER is assumed.

INPLACE
NOTINPLACE is assumed.

ALLOWEMPTY
The server ignores the parameter.

RESOLVE
The server ignores the parameter.

create_date
The server ignores the parameter.

create_time
The server ignores the parameter.

dateref
The server ignores the parameter.

Module:
DMSOPBLK

Severity:
WARNING

98700 (8)

Explanation:
CREATEMIG keyword is not supported by this file pool.

Module:
DMSOPBLK

Severity:
ERROR

98700 (9)

Explanation

Server managing this file does not support certain parameters of DMSOPDBK, and takes these actions when it encounters them:

NORECOVER
RECOVER is assumed.

INPLACE
NOTINPLACE is assumed.

ALLOWEMPTY
The server ignores the parameter.

create_date
The server ignores the parameter.

create_time
The server ignores the parameter.

Module:
DMSOPDBK

Severity:
WARNING

98700 (10)

Explanation

Server managing this file does not support certain parameters of DMSOPEN:

NORECOVER
RECOVER is assumed.

INPLACE
NOTINPLACE is assumed.

ALLOWEMPTY
The server ignores the parameter.

create_date
The server ignores the parameter.

create_time
The server ignores the parameter.

Module:
DMSOPEN

Severity:
WARNING

98700 (11)

Explanation:
The file pool does not support DMSQFPDS.

Module:
DMSQFPDS

Severity:
ERROR

98800

Explanation:
Specified file pool server does not support connections on behalf of user IDs other than the VM ID of the connecting machine: the file pool server is at z/VM Version 1 Release 1.0 or earlier.

Module:
Common

Severity:
ERROR

98900

Explanation:
System error. Unexpected return code or reason code from synchronization point manager routine.

Module:
Common

Severity:
ERROR

99001

Explanation:
Invalid value specified for the *datalen* parameter. It was not in the range of 0 to 2048.

Module:
DMSTRACE

Severity:
ERROR

99002

Explanation:
Invalid value specified for the *id* parameter. It was not in the range of 0 to 65535.

Module:
DMSTRACE

Severity:
ERROR

99003

Explanation:
Invalid value specified for the *machtype* parameter. It was not in the range of 0 to 255.

Module:
DMSTRACE

Severity:
ERROR

99004

Explanation:
The DMSTRACE facility (Monitor Call Class 10) is not enabled.

Module:
DMSTRACE

Severity:
ERROR

99551 (1)

Explanation:
The buffer number is invalid.

Module:
StackBufferDelete—DMSSTKD

Severity:
ERROR

99551 (2)

Explanation:
Invalid buffer number.

Module:
StackQuery—DMSSTKQ

Severity:
ERROR

99551 (3)

Explanation:
The buffer address is incorrect.

Module:
StackRead—DMSSTKR

Severity:
ERROR

99552

Explanation:
Incorrect value for the case parameter.

Module:
StackRead—DMSSTKR

Severity:
ERROR

99553

Explanation:
Incorrect value for *drop_top*.

Module:
StackRead—DMSSTKR

Severity:
ERROR

99554

Explanation:
The value of *line_length* was not in the range 0 to 255.

Module:
StackWrite—DMSSTKW

Severity:
ERROR

99555

Explanation:
Incorrect value for the pad parameter.

Module:
StackRead—DMSSTKR

Severity:
ERROR

99556

Explanation:
Invalid order parameter.

Module:
StackWrite—DMSSTKW

Severity:
ERROR

99557

Explanation:
The program stack is empty.

Module:
StackRead—DMSSTKR

Severity:
ERROR

99558 (1)

Explanation:
There is no buffer with the specified buffer number.

Module:
StackBufferDelete—DMSSTKD

Severity:
ERROR

99558 (2)

Explanation:
There is no buffer with the specified buffer number.

Module:
StackQuery—DMSSTKQ

Severity:
ERROR

99559

Explanation:
The top buffer of the program stack is empty.

Module:
StackRead—DMSSTKR

Severity:
ERROR

99631

Explanation:
Keyword not found in TCPIP DATA file.

Module:
DMSTCD

Severity:
WARNING

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/support/pages/node/6019492), SA22-7935
- Open Systems Adapter-Express Integrated Console Controller User's Guide (https://www.ibm.com/support/pages/node/6019810), SC27-9003
- Open Systems Adapter-Express Integrated Console Controller 3215 Support (https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SA23-2247
- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SC14-7580

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330

- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350033/$file/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350152/$file/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350151/$file/ifc1000_v2r5.pdf), GC35-0151

Related Products

z/OS

- *Common Programming Interface Communications Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf)*, SC26-4399
- z/OS and z/VM: Hardware Configuration Definition Messages (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sc342668/$file/cbdm100_v2r5.pdf), SC34-2668
- z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sc342670/$file/eequ100_v2r5.pdf), SC34-2670
- z/OS: Network Job Entry (NJE) Formats and Protocols (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa320988/$file/hasa600_v2r5.pdf), SA32-0988
- z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa760169/$file/glpa300_v2r5.pdf), SA76-0169
- z/OS: Language Environment Concepts Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380687/$file/ceea800_v2r5.pdf), SA38-0687
- z/OS: Language Environment Debugging Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5ga320908/$file/ceea100_v2r5.pdf), GA32-0908
- z/OS: Language Environment Programming Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380682/$file/ceea200_v2r5.pdf), SA38-0682
- z/OS: Language Environment Programming Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380683/$file/ceea300_v2r5.pdf), SA38-0683
- z/OS: Language Environment Runtime Messages (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380686/$file/ceea900_v2r5.pdf), SA38-0686
- z/OS: Language Environment Writing Interlanguage Communication Applications (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380684/$file/ceea400_v2r5.pdf), SA38-0684
- z/OS: MVS Program Management Advanced Facilities (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa231392/$file/ieab200_v2r5.pdf), SA23-1392
- z/OS: MVS Program Management User's Guide and Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa231393/$file/ieab100_v2r5.pdf), SA23-1393

XL C++ for z/VM

- XL C/C++ for z/VM: Runtime Library Reference, SC09-7624
- XL C/C++ for z/VM: User's Guide, SC09-7625

Index

Special Characters

A

B

C

D

E

F

G

P

Q

R

S

T

U

V

W

Z

Product Number: 5741-A09

Printed in USA

SC24-6259-73